Московский ордена Ленина Государственный Университет
им.М.В.Ломоносова

Механико-Математический факультет

В.А. УСПЕНСКИЙ

ОБЩЕЕ ОПРЕДЕЛЕНИЕ АЛГОРИТМИЧЕСКОЙ ВЫЧИСЛИМОСТИ И АЛГОРИТМИЧЕСКОЙ СВОДИМОСТИ

Дипломная работа

Научный руководитель - академик А.Н.КОЛМОГОРОВ

Май 1952 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

I.

"Алгоритм, алгорифм, - всякая система вычислений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи".[x)] Это определение не является строгим математическим определением, и это не случайно, ибо понятие алгоритма не является чисто-математическим понятием. Задача математики - более или менее адэкватно отобразить это понятие в точных математических терминах.

Существует несколько математических "определений" алгоритма:

A) Определение вычислимой функции как функции, значения которой выводимы в некотором логическом исчислении (Гёдель - Чёрч), см. A.Church [2].

B) Рекурсивные функции Клини (рекурсивные функции определяются через рекурсивные операции), S.C.Kleene [3].

C) Исчисление λ-конверсии Чёрча, A.Church, [2].

D) Вычислительные машины Тьюринга, A.M.Turing [9]

E) Финитный комбинаторный процесс Поста, E.L.Post [4].

F) Нормальные операции Поста, E.L.Post [5], [6].

G) Нормальный алгорифм Маркова, А.А.Марков [I].

Каждое из этих определений задает некоторый специальный вид алгоритма, претендующий на то, что к нему сводит-

----

x) А.Н.Колмогоров, статья "Алгоритм" в БСЭ (2-е издание).

ся любой алгоритм. Конечно, утверждение о сводимости любого алгоритма доказано быть не может (именно в силу недостаточной четкости общего понятия алгоритма). Общность каждого из перечисленных определений, т.е. его способность объять общее понятие алгоритма, устанавливается не строгой теоремой, а более или менее убедительным рассуждением. Сильным доводом в пользу этой общности является также и то, что определения A) – G), повидимому, эквивалентны между собой (эквивалентность A), B), C), D) доказана в работах [2],[10] и др., утверждение об эквивалентности нормального алгорифма другим определениям алгоритма - точнее, утверждение о том, что все известные алгоритмы сводятся к нормальному - ~~высказано без~~ но доказа~~тель~~ В.С. Детловсом [ ] ~~ства А.А.Марковым в [1]~~).

Однако, все эти определения оставляют чувство некоторой неудовлетворенности. Их разумность, т.е. адэкватность общему понятию алгоритма, устанавливается косвенным образом. Сделаем два критических замечания по поводу указанных определений алгоритма.

З а м е ч а н и е I. То́, что́ определяется, не всегда есть алгоритм в том смысле, в каком мы его понимаем. Мы же понимаем алгоритм как функцию, аргументом которой являются входные данные (вопрос, проблема), в некотором закодированном виде, а значением - решение вопроса, проблемы (тоже в закодированном виде). "В математике принято понимать под "алгорифмом" вычивлительный процесс, совершаемый согласно точному предписанию и ве-

дущий от могущих варьировать исходных данных к искомому результату".x)

Разберем с этой точки зрения определение A), представляющееся на первый взгляд наиболее общим. Рассматривается некоторое логическое исчисление, содержащее арифметику, т.е. символы для натуральных чисел и примитивно-рекурсивных функций. Предполагается, что выполняются обычные условия, которым удовлетворяют исчисления такого рода (выводимые формулы образуют вычислимую последовательность и т.п.). Часть этих условий приведена в работе [8]. Такие исчисления мы будем называть допустимыми. Мы назовем теперь функцию, определенную в натуральном ряду, вычислимой, если существует допустимое исчисление F, содержащее символ $\hat{f}$ и такое, что равенство

$$f(m)=n$$

эквивалентно выводимости в F формулы

$$\hat{f}(\hat{m})=\hat{n}$$

где $\hat{m}$ и $\hat{n}$ символы, соответствующие числам m и n. Определение, близкое к этому, дает Чёрч в своей статье [2].

Но это не есть еще алгоритм. Это признает и сам Чёрч, говоря примерно следующее ([2], стр.351):

"Ясно, что для любой вычислимой функции от натурального аргумента существует алгоритм, при помощи кото-

----

x) А.А.Марков, [1].

рого любое частное значение функции может быть эффективно вычислено. Ибо выводимые равенства можно эффективно перенумеровать и алгоритм для вычисления частного значения функции f , обозначаемой символом $\hat{f}$, состоит в просмотре перенумерованного множества выводимых равенств, пока мы не дойдем до требуемого равенства формы $\hat{f}(\hat{m})=\hat{n}$."

Таким образом, определение A) вычислимой функции еще не является определением алгоритма. Понятие вычислимой функции содержит все предпосылки для построения алгоритма, но алгоритмом не является, этот алгоритм x) еще надо строить.

Из конструкции Тьюринга также можно извлечь алгоритм, но сама вычислительная машина задает не алгоритм в нашем понимании (как функцию F , дающую результат при вводе входных данных), а разворачивает множество значений F в вычислимую последовательность.

Эти возражения совершенно неприменимы к определению E) , которое может служить классическим определением некоторого специального типа алгоритма. Однако это определение является именно определением специального типа алгоритма, а не общего алгоритмического процесса. Утверждение об общности определения E) пока что остается бездоказательным (хотя очень вероятно, что оно верно). К определению E) в большей мере, чем к другим определениям, относятся возражения, сформулированные в замечании 2.

---

x) Состоящий: I) в построении последовательности выводимых формул и 2) в просмотре построенных членов последовательности.

З а м е ч а н и е 2. В определениях A) – G) косвенным образом устанавливается достаточная общность определяемого алгоритма. Эта общность обнаруживается

а) путем более или менее неопределенных рассуждений. Этой неопределенности, конечно, избежать полностью нельзя; можно только стремиться сделать рассуждения более убедительными;

б) путем доказательства точных теорем, касающихся эквивалентности определяемого алгоритма и ранее определенных алгоритмов. Справедливость этих теорем не усматривается непосредственно; их доказательство требует специальных рассмотрений. Так, например, Тьюринг, введя в [9] свое определение вычислимой функции, доказывает в [10] эквивалентность этого понятия с понятием общерекурсивной и λ-определимой функции так:

1) ссылается на результат Клини, доказавшего, что каждая общерекурсивная функция λ-определима,

2) вводит понятие λ-K-определимости и утверждает, что каждая λ-определимая функция λ-K-определима,

3) доказывает, что каждая λ-K-определимая функция вычислима по Тьюрингу,

4) доказывает, что каждая вычислимая по Тьюрингу функция общерекурсивна.

Эти же возражения относятся и к определениям C), E), F), G).

2.

Все вышесказанное имело своей целью обосновать целесообразность введения нового, более совершенного определения алгоритма, к которому мы предъявим, таким образом, следующие два требования:

I. Это должен быть действительно алгоритм.

2. Этот алгоритм должен быть достаточно общим. Причем желательно, чтобы эта общность устанавливалась не косвенным образом, путем специальных рассмотрений, а по возможности содержалась в самом определении.

Такое определение предложил А.Н.Колмогоров. Чтобы подойти к этому определению, попытаемся уловить наиболее существенные черты, свойственные самому общему алгоритмическому процессу.

Вычисления, о которых идет речь в определении алгоритма, могут производиться либо на бумаге (например, нормальный алгорифм Маркова), либо механически (например, вычислительная машина Тьюринга). Состояние процесса в каждый момент времени определяется в первом случае - конфигурацией символов на бумаге, во втором случае - конфигурацией звеньев машины. Каждая такая конфигурация имеет следующую структуру: имеется фиксированный конечный запас "элементов" (символов, звеньев машины); конфигурация состоит из этих элементов (каждый элемент может встречаться в этой конфигурации более одного раза); между некоторыми из элементов имеется "связь". Так, в нормальном алгорифме Марко-

ва такой конфигурацией является слово, "элементами" - буквы алфавита, каждая буква "связана" с соседней буквой слева и соседней буквой справа. В вычислительной машине Тьюринга конфигурацией является состояние машины; "элементами" - символы, напечатанные на ленте, и состояние командного устройства; "связанными" следует считать символы, напечатанные на соседних секциях ленты, кроме того состояние командного устройства связано с символом воспринимаемой секции.

Элементы конфигурации будем теперь изображать точками (вершинами), с указанием при каждой точке обозначения соответствующего элемента, в связи с элементами - отрезками, соединяющими эти точки.[x)] Тогда вся конфигурация изобразится одномерным топологическим комплексом с заданной на его вершинах функцией f , принимающей значения из нашего запаса элементов.

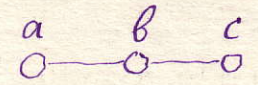
Сделаем одно уточнение. От каждого элемента отходит, вообще говоря, несколько связей, или, что то же самое, от каждой вершины отходит несколько отрезков. Иногда бывает нужным упорядочить эти связи (отрезки). Так, например, в нормальном алгорифме Маркова от каждой буквы отходит две связи - одна к левой соседней букве и другая к правой соседней букве; при этом существенно эти связи различать. В соответствующем одномерном комплексе мы можем достичь этого следующим образом. Пусть слева от буквы b стоит буква a , а справа - буква c . Соответствующая часть

---

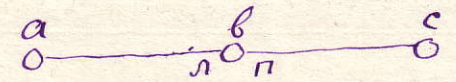
x) Например, слово $abcd$ из нормального алгорифма Маркова изобразится так:



одномерного комплекса будет иметь вид:



Поставим на отрезке ba вблизи буквы b значок "л" (левая связь), а на отрезке bc вблизи буквы b значок "п" (правая связь). Получим схему



Проделаем эту операцию для каждой вершины. Каждый отрезок получит два значка. Слово $abcd$ изобразится теперь в виде



В общем случае каждый отрезок, отходящий от некоторой вершины, получит значок, стоящий вблизи этой вершины; каждый отрезок таким образом получит два значка (по одному на каждом из своих концов). При этом требуется, чтобы значки отрезков, отходящих от любой вершины, стоящие вблизи этой вершины, были различны. Значки будем выбирать из некоторого заранее установленного конечного запаса значков.

Перенумеруем теперь наш конечный запас элементов натуральными числами I, 2, ..., n , а конечный запас значков - натуральными числами I, 2, ..., α и отождествим элементы и значки с их номерами. Итак, состояние алгоритмического процесса есть одномерный топологический комплекс с заданной на его вершинах характеристической функ-

цией f , принимающей значения I,2, ..., n . Каждому концу каждого отрезка отнесен значок из множества номеров I, 2, ...,α; при этом отрезки, сходящиеся в любой вершине, несут на концах, обращенных к этой вершине, различные значки. Одномерный комплекс указанного типа мы будем называть одномерным комплексом порядка (n,α), а чаще всего просто комплексом (см.§ I).

Проблема задается в виде комплекса; решение также получается в виде комплекса.

Алгоритмический процесс производится шагами. Каждый шаг заключается в переработке одного комплекса в другой по определенным правилам переработки. Алгоритм - процесс детерминированный, поэтому комплекс, получившийся на n -ом шагу, однозначно определяет комплекс, который получится на ($n+1$)-м шагу. Однако каждый раз мы в состоянии воспринять не весь комплекс, содержащий, вообще говоря, сколь угодно много вершин, а лишь некоторую его "обозримую" часть (причем объем обозримой части не может превосходить заранее установленного предела). Правила переработки должны быть таковы, что переработка происходит исключительно на основании информации о виде обозримой части и затрагивает только обозримую часть.

С другой стороны, мы должны в принципе уметь воспользоваться всей информацией, содержащейся в комплексе, а не только той, которая попала в его обозримую часть. Поэтому мы должны обеспечить обозримой части возможность передвигаться по комплексу. Таким образом, после преобразования обозримой части она, эта обозримая часть, долж-

на передвинуться, т.е. некоторые "необозримые" вершины должны стать обозримыми (и некоторые обозримые - необозримыми [x]).

Итак, пусть на n-м шагу процесса возник комплекс K^n. Его надо преобразовать в K^{n+1}. Это делается в два приема:

1) перестраивается обозримая часть комплекса K^n,

2) после этого некоторые необозримые вершины объявляются обозримыми (а некоторые обозримые - необозримыми).

Но как указать те необозримые вершины, которые становятся обозримыми (назовем их потенциально-обозримыми)? Как описать их, когда они лежат за пределами обозримой части? У нас нет никаких других ориентиров, кроме обозримых вершин. Потенциально-обозримые вершины мы можем описать лишь отправляясь от обозримых; потенциально-обозримые вершины должны быть связаны с обозримыми короче говоря, потенциально-обозримые вершины должны находиться в замыкании $[A]$ множества A обозримых вершин (определение замыкания см. § I, стр. 34).

Рассмотрим все это несколько подробнее. Обозримая часть состоит:

1) из множества A обозримых вершин,

2) из множества обозримых отрезков, соединяющих между собой обозримые вершины,

3) из множества "полуобозримых" отрезков, соединяющих обозримые вершины с необозримыми. Мы назвали их "по-

----

[x] Обозримые вершины - принадлежащие к обозримой части, необозримые - остальные.

лубобозримыми", ибо в них обозрим только тот конец, который обращен к обозримой вершине, т.е. из двух значков, стоящих на этом отрезке, в обозримую часть входит только один - на конце, обращенном к обозримой вершине.

Мы видим, что обозримая часть образует в комплексе то, что мы в § I назовем подклассом. Этот подкласс (обозначим его $\mathfrak{A}$) натянут на множество A обозримых вершин; полуобозримые отрезки являются внешними для этого подкласса.

Таким образом, состояние процесса есть комплекс K с выделенным в нем подклассом, который объявлен обозримой частью.

На основе вида обозримой части $\mathfrak{A}$ происходит ее переработка. Чтобы не нарушать связи между обозримой и необозримой частями, потребуем, чтобы крайние x) вершины $\mathfrak{A}$ оставались неподвижными (хотя характеристическая функция на этих вершинах может и измениться); также должны остаться неизменными полуобозримые отрезки (хотя, опять-таки, обозримые значки на них могут измениться).

Итак, подкласс $\mathfrak{A}$ преобразовался в подкласс $\mathfrak{A}^*$, причем крайние вершины остались неподвижными (через них - при помощи полуобозримых отрезков - $\mathfrak{A}^*$ сообщается с необозримой частью). Теперь надо выделить новое множество обозримых вершин. Оно выделяется в замыкании $[\mathfrak{A}^*]$ подкласса $\mathfrak{A}^*$. Те вершины, которые выделяются в самом $\mathfrak{A}^*$, указываются непосредственно. Те вершины, которые выделяются в разности $[\mathfrak{A}^*] - \mathfrak{A}^*$, выделяются

----

x) Определение крайних вершин см. § I, стр. 34

путем указания тех полуобозримых отрезков, которые к ним подводят.[x)] Таким образом возникает новое множество обозримых вершин A' и натянутый на него подкласс $\mathfrak{A}'$.

Итак, на n-м шагу имеется комплекс K^n и в нем обозримая часть - подкласс $\mathfrak{A}$. На основании информации о виде $\mathfrak{A}$ происходит переработка K^n в K^{n+1}; переработка происходит следующим образом: подкласс $\mathfrak{A}$ заменяется подклассом $\mathfrak{A}^*$, сохраняющим крайние вершины (и внешние отрезки), затем в пределах $[\mathfrak{A}^*]$ выделяется новый подкласс $\mathfrak{A}'$, который объявляется обозримой частью.

Мы не испортим дела, если несколько обобщим процесс переработки K^n в K^{n+1}, не нарушая его эффективности. Именно, будем считать, что переработка происходит на основании информации о виде не только $\mathfrak{A}$, но всего замыкания $H=[\mathfrak{A}]$; переработка затрагивает не только $\mathfrak{A}$, но весь подкомплекс H, который перерабатывается в H^*; при этом не будем требовать неподвижности крайних вершин $\mathfrak{A}$, а потребуем неподвижности вершин, соседних для $\mathfrak{A}$, т.е. вершин из разности $H \setminus \mathfrak{A}$; посредством этих вершин H^* приклеивается к остальной, не изменившейся, части комплекса. После замены H на H^* в пределах H^* происходит выделение нового множества A'-обозримых вершин. Замену H на H^* и выделение в H^* множества A' мы можем объединить в один шаг. Для этого будем считать,

----

x) Каждый полуобозримый отрезок выделяется указанием обзримой вершины, от которой он отходит, и значка, стоящего на этом отрезке вблизи этой вершины.

что внутри H^* уже выделено множество A'. Поэтому шаг состоит просто в замене подкомплекса H на подкомплекс H^*.

Условимся считать, что алгоритм производится некоторой машиной, перерабатывающей комплексы. У нее, как у всякой машины, есть некоторое начальное состояние (до ввода начальных данных) - тоже в виде комплекса. Входные данные в виде комплекса присоединяются к начальному состоянию - получается комплекс K^0. По нашим правилам он перерабатывается в K^1, K^1 перерабатывается в K^2 и т.д. до тех пор, пока мы не получим сигнал о конце процесса, т.е.о получении решения.

Комплексы K^0, K^1, K^2 и т.д. не просто комплексы: в каждом из них выделено множество обозримых вершин; в дальнейшем нам будет удобнее называть эти вершины активными. Переработка происходит на основе информации о виде подкомплекса $H=[A]$, который мы назовем потенциальным (а его вершины - потенциальными). Вершины из разности $H \setminus A$ назовем граничными. Потенциальные вершины таким образом делятся на активные и граничные. Все остальные (не потенциальные) вершины назовем пассивными. Комплекс с выделенным в нем множеством активных вершин назовем препарированным комплексом или П-комплексом. Обыкновенный комплекс можно считать частным случаем П-комплекса, если положить множество активных вершин пустым.

Приведенные рассуждения подводят нас к следующему определению алгоритма, предложенному А.Н.Колмогоровым.

1. Машина перерабатывает П-комплексы с единственной активной вершиной в П-комплексы с единственной активной вершиной.*)

2. Машина задается начальным состоянием и правилами переработки.

Начальное состояние есть конечный или бесконечный (но ограниченный - см.§ 1) П-комплекс с единственной активной вершиной.

Правила переработки состоят из конечного множества $\mathfrak{M}$ упорядоченных пар $\{(H, H^*)\}$.

Во всякой паре первый элемент H есть конечный П-комплекс, являющийся замыканием множества своих активных вершин, а второй элемент H^* есть или конечный П-комплекс или слово "стоп".

Если H^* есть "стоп", то соответствующий П-комплекс H имеет единственную активную вершину.

Если H^* есть П-комплекс, то задано взаимно-однозначное соответствие между граничными вершинами H и некоторым множеством вершин H^*.

Не может быть двух пар, у которых первые элементы одинаковы, а вторые различны.

3. Машина работает шагами. Каждый шаг состоит либо в замене возникшего перед этим шагом П-комплекса K на П-комплекс K', либо в получении сигнала о решении, либо в безрезультатной остановке.

Замена K на K' происходит следующим образом: в K берется потенциальный подкомплекс H_K и среди пар $\mathfrak{M}$ ищется та, которая своим первым элементом имеет H_K.

---

*) Промежуточные П-комплексы, возникающие в процессе переработки, могут иметь и более одной активной вершины.

Если второй элемент H^* в этой паре есть П-комплекс, то $H_к$ заменяется (внутри K) на H^*. Отрезки, соединявшие пассивные вершины П-комплекса K x) с граничными, подводятся теперь к соответствующим вершинам H^*. Полученный таким образом комплекс и есть K'.

Если второй элемент пары, первым элементом которой является $H_к$, есть слово "стоп", то работа машины прекращается; связная компонента Q единственной активной вершины K есть решение.

Если среди пар $\mathfrak{M}$ нет пары, первый элемент которой есть $H_к$, то машина останавливается безрезультатно.

4. Подлежащий переработке комплекс P присоединяется своей активной вершиной к активной вершине начального состояния, что дает комплекс K^0. После этого машина начинает работать, заменяя K^0 на $K^1=(K^0)'$, K^1 на $K^2=(K^1)'$ и т.д. до получения решения или безрезультатной остановки.

В Приложении (стр. 87) работа машины А. Н. Колмогорова демонстрируется на простом примере.

Машина А.Н.Колмогорова определяет в множестве П-комплексов некоторую функцию $Q=\Gamma(K)$. Функцию от П-комплексов, задаваемую некоторой машиной Колмогорова, будем называть алгоритмической.

Алгоритмическая функция Γ не всюду определена. Действительно, П-комплексы, участвующие в задании машины: начальное состояние и элементы пар множества

---

x) В смысле взаимно-однозначного соответствия, установленного между граничными вершинами $H_к$ и некоторым множеством вершин H^*.

$\mathfrak{M}$ - ограничены. Всех их конечное число, следовательно, их порядки не превосходят некоторого порядка (n,α) (см.§ I). Поэтому функция Γ может быть определена только для тех комплексов, у которых в процессе переработки потенциальные подкомплексы будут иметь порядки не выше (n,α). Естественно поэтому рассматривать функцию Γ только на П-комплексах порядка не выше (n,α).

Каждой машине мы припишем порядок - минимальный порядок (n,α) такой, что порядки всех П-комплексов, участвующих в задании машины, не превосходят (n,α). Машина порядка (n,α) задает алгоритмическую функцию Γ от комплексов порядка не выше (n,α).

Но дело не в этом. Даже и на комплексах порядка не выше (n,α) функция Γ является, вообще говоря, не всюду определенной. Ибо процесс переработки комплекса P может не иметь результативного окончания:

а) потому, что процесс может не окончиться,

б) потому, что машина может остановиться безрезультатно.

При этом не существует эффективного общего метода (алгоритма), позволяющего узнавать по заданной алгоритмической функции Γ и заданному П-комплексу P, определена ли функция $\Gamma(P)$ или нет (или, как говорят, применим ли алгоритм Γ к П-комплексу P или нет). Этот факт носит общий характер, он остается в силе для всех определений алгоритма.x)

---

x) Более того, всегда существует такой фиксированный алгоритм $\mathcal{F}_0$, что не существует никакого алгоритма, позволяющего для любых входных данных P указать, применим ли $\mathcal{F}_0$ к P или нет.

Таким образом то, что мы определили, следовало бы назвать скорее частичным алгоритмом. Однако, все другие определения алгоритма тоже дают только частичный алгоритм, да ничего другого дать и не могут (не частичный алгоритм неизбежно определяется как такой частичный алгоритм, который применим ко всяким входным данным[x)]). Поэтому мы отбросим эпитет "частичный" и будем то, что мы определили, называть просто "алгоритмом".

Назовем алгоритм безусловным, если начальное состояние пусто, и условным в противном случае.

3.

Все другие определения алгоритма, точнее, алгоритмы в смысле других определений, автоматически записываются в терминах безусловного алгоритма Колмогорова. Хочется при этом подчеркнуть, что речь идет не о сводимости определенного класса алгоритмов, скажем нормальных алгорифмов Маркова, к алгоритму Колмогорова, а именно об автоматическом переводе алгоритмов этого класса, в частности нормальных алгорифмов Маркова, на язык алгоритмов Колмогорова (см.§ 2). Это происходит в силу уже отмеченного обстоятельства, что каждый алгоритм является по существу процессом, производимым над комплексами, причем процессом как раз того типа, который описывается определением А.Н.Колмогорова.

----

x) Например, общерекурсивная функция определяется как такая частично рекурсивная, которая определена на всем натуральном ряду.

Определение рекурсивной функции, как функции, заданной последовательностью рекурсивных равенств, не есть еще определение алгоритма. Но если отсюда извлечь алгоритм, т.е. совершенно точно указать последовательность действий, позволяющих по этим равенствам найти значение функции, то тем самым автоматически получится некоторый алгоритм Колмогорова. Активные вершины будем обозначать, обводя их красным кружком. Для каждой частично-рекурсивной функции

$$y = \varphi(x_1, x_2, \ldots, x_n)$$

легко построить алгоритм (см.§ 2), переводящий П-комплекс x)

(1) $\varphi\ (\ 2\ 2 \ldots 2\ 2\ ,\ 2\ 2 \ldots 2\ 2\ , \ldots ,\ 2\ 2 \ldots 2\ 2\ )$ — x_1 раз, x_2 раз, x_n раз

в П-комплекс $2\ 2\ 2 \ldots 2$ — y раз

(2)

x) До сих пор мы для простоты предполагали, что характеристическая функция f принимает значения из натурального ряда. Натуральные числа при этом играли просто роль символов. Ничего не изменится, если разрешить характеристической функции принимать значения из любого конечного или пересчитанного бесконечного множества символов, например содержащего знаки ( ) , φ Ψ и т.д. (Ведь сами натуральные числа получились у нас в результате занумерования некоторого множества символов).

Иногда, как это мы сделали для П-комплексов (1) и (2), мы будем, изображая П-комплексы на бумаге, опускать значки, поставленные на концах отрезков.

При всей своей общности алгоритм Колмогорова, как и следовало ожидать, оказывается не шире, чем обычные рекурсивные функции. Каждому П-комплексу K можно эффективно и однозначно сопоставить натуральное число k - его номер (так, чтобы по номеру эффективно и однозначно восстанавливался комплекс).[x)] Функция от комплексов $L = \Gamma(K)$ (вообще говоря, не всюду определенная) индуцирует в натуральном ряду функцию $l = \gamma(k)$ (тоже, вообще говоря, не всюду определенную).[xx)]

Теорема. Если $\Gamma(K)$ - алгоритмическая функция, то $\gamma(k)$ - частично-рекурсивная функция (§ 3).

Обратное, однако, неверно. Не для всякой частично-рекурсивной функции $\gamma(k)$ существует соответствующая

---

x) Комплексы будем обозначать большими латинскими буквами, их номера - соответствующими малыми.

xx) Функции от комплексов мы будем обозначать большими греческими буквами, а индуцированные функции в натуральном ряду - соответствующими малыми буквами.

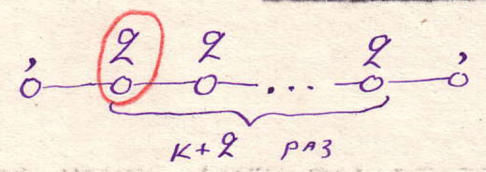
Рассматривая функцию от элементов некоторого множества Γ (комплексов, натуральных чисел и т.д.), мы разрешаем ей быть не всюду определенной. Про такую функцию мы скажем, что она определена в множестве Γ. В дальнейшем без оговорок рассматриваются функции, определенные в некотором множестве. Равенство двух функций

$$a(x) = b(x)$$

мы будем понимать в том смысле, что обе функции одновременно определены или не определены и их значения равны.

алгоритмическая функция $\Gamma(K)$. Например, если $\gamma(k)$ определена на всем множестве номеров комплексов, то $\Gamma(K)$ должна быть применима ко всем комплексам, в то время как алгоритмическая функция не может быть применима ко всем комплексам.

Тем не менее, для каждой функции в натуральном ряду $\gamma(k)$ все же можно найти ее представитель среди функций от комплексов. Для этого будем каждое натуральное число k представлять его <u>изображением</u> - П-комплексом $\overline{K}$:



Натуральные числа будем изображать малыми латинскими буквами, их изображения - соответствующими большими буквами с чертой наверху. Для каждой функции $l = \gamma(k)$ в натуральном ряду существует функция от комплексов - мы обозначим ее соответствующей большой греческой буквой с чертой наверху - $\overline{L} = \overline{\Gamma}(\overline{K})$

<u>Теорема</u>. Если $\gamma(k)$ частично-рекурсивная функция, то $\overline{\Gamma}(\overline{K})$ - алгоритмическая функция. (§ 2).

Назовем функцию $l = \gamma(k)$ вычислимой, если существует алгоритмическая функция $\overline{\Gamma}$, такая что $\overline{L} = \overline{\Gamma}(\overline{K})$ Легко видеть (§ 3), что <u>класс вычислимых функций совпадает с классом частично-рекурсивных</u>.

Здесь и всюду во "Введении" речь для простоты идет о целочисленных функциях только от одного аргумента. Однако все факты сохраняют свою силу и для функций от любого числа аргументов.

4.

Перейдем к рассмотрению условных алгоритмов. Если начальное состояние машины, задающей условный алгоритм, есть конечный П-комплекс, то такую машину легко заменить машиной, определяющей ту же функцию в множестве комплексов, но с пустым начальным состоянием. Такой алгоритм является по существу безусловным.

При рассмотрении условных алгоритмов поэтому нас будут интересовать машины с бесконечным начальным состоянием. Мы оставим в стороне общий случай и займемся машинами с начальным состоянием специального вида, который укажем позже.

Вопрос об условных алгоритмах мы рассмотрим в связи с проблемой сводимости.

Наряду с проблемой решения отдельной задачи существует проблема алгоритмического решения серии задач. Точно так же наряду с проблемой сведения решения одной задачи к решению другой задачи существует проблема алгоритмического сведения одной серии задач к другой серии задач.

Проблемой сводимости занимался Поуст [6]. Он рассматривал два рекурсивно перечислимых, но не рекурсивных множества натуральных чисел S_1 и S_2. Первая серия задач: "Принадлежит ли n множеству S_1". Вторая серия задач: "Принадлежит ли m множеству S_2". Поуст рассматривал вопрос об алгоритмическом сведении первой серии ко второй. Это есть проблема сводимости множества S_1 к множеству S_2.

Сводимость множеств есть частный случай сводимости функций (при сводимости S_1 к S_2 характеристическая функция S_1 сводится к характеристической функции S_2). Мы будем изучать проблему сводимости сразу для функций.

Как определить сводимость функции $\gamma(n)$ к функции $\delta(n)$? Что означает утверждение: "функция $\gamma(n)$ вычислима, при условии, что вычислима $\delta(n)$ "? Проще всего сказать, что это означает следующее положение: "если функция $\delta(n)$ вычислима, то и $\gamma(n)$ вычислима. Но отсюда получается, что все невычислимые функции сводимы друг к другу просто в силу ложности посылки. Значит, надо дать какое-то другое, более тонкое определение. Таких определений можно предложить два (не касаясь определений некоторых частных случаев сводимости, рассмотренных Постом в [6]):

I) Рекурсивная сводимость. Функция $\gamma(n)$ рекурсивно сводится к функции $\delta(n)$, если $\gamma(n)$ принадлежит рекурсивному замыканию x) $\delta(n)$. Идея такого определения принадлежит Б.А.Трахтенброту.

2) Тьюринговская сводимость. Понятие тьюринговской сводимости было введено Постом в [6] :

"Эффективное решение проблемы разрешимости для рекурсивно-перечислимого множества S_1 натуральных чисел

---

x) Рекурсивное замыкание функции δ есть минимальный рекурсивно-замкнутый класс, содержащий δ и все примитивно-рекурсивные функции. Класс функций называется рекурсивно-замкнутым, если он замкнут относительно рекурсивных операций: суперпозиций, примитивных рекурсий и применений оператора μ .

может быть мыслимо в виде машины, или последовательности правил, которая, если задать любое натуральное число развернет ~~моногенный~~ однородный (monogenic) процесс, оканчивающийся правильным ответом "да" или "нет" на вопрос "принадлежит ли n к S_1 ". Предположим теперь, что эта ситуация имеет место со следующей модификацией. Пусть в определенное время другой машинно-задаваемый процесс ставит вопрос, принадлежит ли определенное число m данному рекурсивно-перечислимому множеству S_2 , и пусть машина так устроена, что если правильно отвечать на этот вопрос всякий раз, как вопрос возникает, то процесс будет автоматически продолжаться до окончательного ответа (на вопрос " $n \in S_1$? " - В.У.). Мы можем тогда сказать, что машина эффективно сводит проблему разрешимости для S_1 к проблеме разрешимости для S_2 . Интуитивно это соответствует наиболее общей концепции сводимости S_1 к S_2. Потому что сама концепция проблемы разрешимости для S_2 содержит в себе единственно "отвечание" для любого данного натурального числа m на вопрос, содержится ли m в S_2 , а за конечное время конечное число таких вопросов может быть задано. Соответствующая формулировка "тьюринговской сводимости" имеет поэтому ту же степень общности по отношению к эффективной сводимости, как, скажем, общерекурсивные функции - по отношению к эффективной вычислимости. Отметим, что ... в тьюринговской сводимости, за исключением первого числа m , значения чисел m , для которых этот вопрос (вопрос " $m \in S_2$? " - В.У.) ставится, зависит, вообще говоря,

от правильных ответов на эти вопросы для всех предшествующих m. Характер этой зависимости, однако, эффективен, и мы имеем эффективную сводимость в интуитивном смысле" (Пост, [6], стр.311-312).

Формулировка Поста, таким образом, касается сводимости множеств, или, что то же самое, характеристических функций. Однако, она без труда может быть перенесена на сводимость произвольных функций в натуральном ряду (§ 4).

В § 4 показана эквивалентность этих обоих определений сводимости.

Попытаемся теперь дать наиболее общее определение алгоритмической сводимости решения одной серии проблем к решению другой серии проблем в терминах алгоритма Колмогорова, а именно условного алгоритма.

Каждую проблему будем задавать в виде П-комплекса K с одной активной вершиной. Решение проблемы есть П-комплекс L также с одной активной вершиной. Очевидно, L есть функция от K:

$$L = \Lambda(K)$$

Пусть первая серия проблем образует множество П-комплексов $\{B\}$, их решения образуют множество П-комплексов $\{C\}$, $C = \Gamma(B)$. Аналогично для второй серии проблем - множества $\{P\}$ и $\{Q\}$ и функция $Q = \Delta(B)$. Речь идет, следовательно, о том, чтобы алгоритмически свести функцию Γ к функции Δ.

При изучении функций от комплексов наложим на эти функции следующее ограничение. Назовем последовательность

П-комплексов

$$S_1, S_2, \ldots, S_m, \ldots$$

перечислимой, если существует безусловный алгоритм Φ, такой что

$$\Phi(\overline{M}) = S_m$$

(напомним, что $\overline{M}$ есть изображение числа m). Множество П-комплексов называется перечислимым, если его можно расположить в перечислимую последовательность (быть может, с повторениями). Множество П-комплексов, к которым применима произвольная алгоритмическая функция, всегда перечислимо (§ 5). Поэтому в дальнейшем мы будем рассматривать исключительно функции от комплексов, определенные на перечислимых множествах. Что же касается функций в натуральном ряду, то мы будем рассматривать только функции, области определения которых рекурсивно-перечислимы (таковы все частично-рекурсивные функции).

Вернемся к вопросу о сводимости функции Γ к функции Δ. Построим бесконечный П-комплекс ξ, заключающий в себе всю информацию о функции Δ. Для этого развернем область определения Δ в перечислимую последовательность:

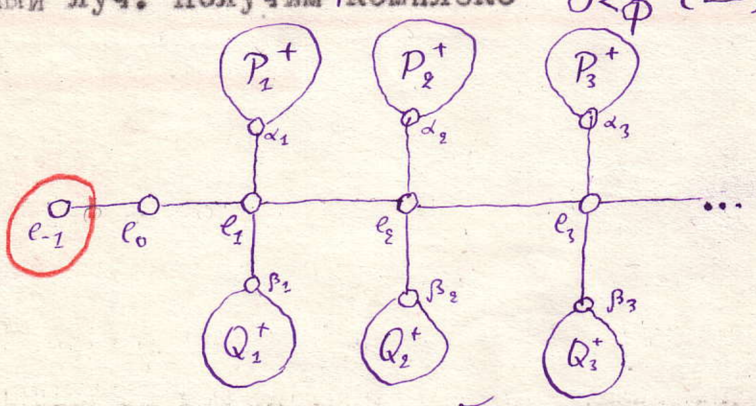
$$P_1, P_2, \ldots, P_m, \ldots \qquad P_m = \Phi(\overline{M})$$

Множество значений Δ тоже развернем в последовательность (вообще говоря, неперечислимую):

$$Q_1, Q_2, \ldots, Q_m, \ldots \qquad Q_m = \Delta(P_m)$$

Каждый П-комплекс P_i имеет единственную активную вер-

шину α_i. Заменим P_i на комплекс P_i^+, отличающийся от P_i лишь тем, что в P_i^+ нет активных вершин (т.е. все вершины P_i, в том числе и α_i, в P_i^+ присутствуют, но ни α_i и никакая другая вершина не выделена как активная.) П-комплекс Q_i имеет единственную активную вершину β_i; совершенно аналогично строится Q_i^+. Соединим вершину α_i комплекса P_i^+ и вершину β_i комплекса Q_i^+ отрезком; посредине этого отрезка поставим вершину e_i. Все эти вершины e_i нанижем на бесконечный луч. Получим П-комплекс $К_\Phi(\Delta)$:



Активной вершиной в $К_\Phi(\Delta)$ является e_{-1}. Для того, чтобы аккуратно получить П-комплекс, надо еще в $К_\Phi(\Delta)$ задать характеристическую функцию на вершинах и значки на концах отрезков. Характеристическая функция полагается равной единице на вершинах e_i и той, которая была, на вершинах P_i^+ и Q_i^+. Аналогичным образом устраивается и распределение значков на концах отрезков.



Мы скажем, что функция Γ алгоритмически сводится к функции Δ, если существует такая алгоритмическая функция Φ и условный алгоритм Π_Φ с начальным состоянием $К_\Phi(\Delta)$, что для любого комплекса B

$$\Pi_\Phi(B) = \Gamma(B)$$

Про условный алгоритм Π_φ мы скажем в таком случае, что он сводит Γ к Δ. Пусть φ и θ - две алгоритмические функции, каждая из которых развертывает область определения Δ в перечисленную последовательность. Можно показать, что если существует алгоритм Π_φ, сводящий Γ к Δ, то существует и алгоритм Π_θ, сводящий Γ к Δ, так что сам факт сводимости не зависит от выбора функции φ (§ 5).

Установим связь алгоритмической сводимости функций от комплексов с рекурсивной сводимостью функций от натуральных чисел.

Пусть даны две функции от комплексов $\Gamma(K)$ и $\Delta(K)$. Они (см.стр. 19) индуцируют в натуральном ряду функции $\gamma(k)$ и $\delta(k)$. Функциям $\gamma(k)$ и $\delta(k)$ в свою очередь отвечают функции $\bar{\Gamma}(\bar{K})$ и $\bar{\Delta}(\bar{K})$ (см.стр. 20).

Лемма I. Если $\Gamma(K)$ алгоритмически сводится к $\Delta(K)$, то $\gamma(k)$ рекурсивно сводится к $\delta(k)$.

Лемма 2. Если $\gamma(k)$ рекурсивно сводится к $\delta(k)$, то $\bar{\Gamma}(\bar{K})$ алгоритмически сводится к $\bar{\Delta}(\bar{K})$.

Отсюда нетрудно получить теорему

Для того, чтобы $\gamma(k)$ рекурсивно сводилась к $\delta(k)$, необходимо и достаточно, чтобы $\bar{\Gamma}(\bar{K})$ алгоритмически сводилась к $\bar{\Delta}(\bar{K})$.

Попутно обнаруживается следующее обстоятельство. Частично-рекурсивная функция определяется как элемент рекурсивного замыкания класса примитивно-рекурсивных функций, после чего доказывается теорема о каноническом представлении каждой частично-рекурсивной функции в виде:

$$f(x) = \varkappa(\mu y[\theta(x,y) = 0])$$

где $\varkappa$ и θ - примитивно-рекурсивные функции, причем $\varkappa$ - некоторая вполне определенная функция (Kleene, [3]). Частично-рекурсивная функция, таким образом, определяется как результат какого-то числа рекурсивных операций; после чего доказывается, что можно обойтись вполне определенным числом операций, из которых две: μ и $\varkappa$ - раз навсегда фиксированы. Точно так же и для произвольной функции γ, сводящейся к функции δ, можно написать некоторое каноническое выражение. По предположению область M определения функции δ рекурсивно-перечислима, т.е. можно построить такую общерекурсивную функцию $\varphi(m)$, что $\{\varphi(m)\} = M$. Известно, что в качестве $\varphi(m)$ всегда можно взять некоторую примитивно-рекурсивную функцию.

Теорема. Существуют две вполне определенные примитивно-рекурсивные функции $\tau(u)$ и $\omega(u)$ со следующими свойствами. Если $\gamma(x)$ рекурсивно сводится к $\delta(x)$, то существует примитивно-рекурсивная функция $h(u,v,w)$ такая, что выполняются равенства

I) $g(x,0) = x$

$g(x, m+1) = h(g(x,m), \varphi(m), \delta(\varphi(m)))$

2) $\gamma(x) = \tau(g(x, \mu m[\omega(g(x,m)) = 0]))$

где $\varphi(m)$ - примитивно-рекурсивная функция, пересчитывающая область определения $\delta(x)$.

Таким образом, функция γ называется сводящейся к δ, если она получается из δ и примитивно-рекурсивных функций произвольным числом рекурсивных операций: оказывается, что можно ограничиться вполне определенным числом операций, из которых три: μ, τ и ω - фиксированы.

В частном случае, когда δ всюду определена, равенства I/ - 2/ могут быть переписаны в виде

I/ $g(x,0) = x$

$g(x,m+1) = h(g(x,m), m, \delta(m))$

2/ $\gamma(x) = \tau(g(x, \mu m[\omega(g(x,m)) = 0]))$

\- - - - -

В §§ I-5 более подробно рассматривается то́, о чём говорилось во "Введении". В § I даются строгие определения комплекса, подкомплекса, подкласса и т.д. В § 2 показывается, как записать в терминах алгоритма Колмогорова нормальный алгорифм Маркова и алгоритм рекурсивных функций. В § 3 доказывается, что каждая алгоритмическая функция индуцирует в натуральном ряду частично-рекурсивную функцию. В § 4 рассматривается проблема сводимости функций. В § 5 устанавливается ряд фактов, упомянутых в предыдущих параграфах, говорится об универсальном алгоритме и т.д.

Андрею Николаевичу Колмогорову автор обязан как постановкой задач, так и постоянным руководством во время выполнения этой работы. Автор приносит Андрею Николаевичу свою глубокую благодарность.

ОБОЗНАЧЕНИЯ

Натуральные числа обозначаются обычно малыми латинскими буквами:

$k, l, m, n, p, q, x, y, z$ и т.д.

П-комплексы обозначаются большими латинскими буквами:

$K, L, M, N, P, Q, X, Y, Z$ и т.д.

Функции от натуральных чисел обозначаются обычно малыми греческими буквами:

$\gamma(k), \delta(x), \varphi(m)$ и т.д.

Функции от комплексов обозначаются большими греческими буквами:

$\Gamma(K), \Delta(X), \Phi(M)$ и т.д.

Иногда к этим буквам приписываются значки и индексы:

$\bar{x}, k_1, s_r, \bar{X}, K_1, S_r, \bar{\Delta}$ и т.д.

При этом соблюдаются следующие условия:

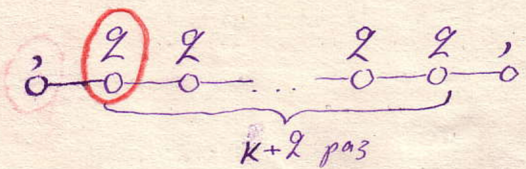
1) Если большая латинская буква обозначает П-комплекс, то соответствующая малая буква обозначает номер этого комплекса. Так, комплекс K имеет номер k.

2) Если большая греческая буква обозначает функцию от комплексов, то соответствующая малая буква обозначает индуцированную функцию от номеров комплексов. Так, функция $Y=\Phi(X)$ индуцирует в натуральном ряду функцию

$y=\varphi(x)$.

3) Если малая латинская буква (например, k) обозначает натуральное число, то соответствующая большая латинская буква с чертой наверху $\bar{K}$ обозначает П-комп-

лекс, являющийся изображением этого числа, т.е. П-комплекс



4) Если малая греческая буква (например, φ) обозначает функцию в натуральном ряду, то соответствующая большая греческая буква с чертой наверху обозначает <u>изображение</u> этой функции, т.е. функцию от комплексов $\overline{\Phi}$, такую что если $y=\varphi(x)$, то $\overline{Y}=\overline{\Phi}(\overline{X})$

Например, номер комплекса M есть m , изображение этого номера - комплекс $\overline{M}$, номер этого комплекса $\overline{m}$ и т.д.

Следующие <u>фиксированные</u> примитивно-рекурсивные функции имеют постоянные обозначения:

$\omega(x)$ — функция, равенство нулю которой эквивалентно тому, что на одной из вершин П-комплекса X с номером x имеется знак ω.

$\nu(x_1, x_2, \ldots, x_n)$ — номер изображения "энки" чисел $(x_1, x_2, \ldots, x_n)$ (т.е. номер П-комплекса $(\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_n)$:
(x_1+2 раз … x_2+2 раз … x_n+2 раз)

$\zeta(u)$ — функция, такая что если u есть номер П-комплекса $\overline{X}$ - изображения числа x, то $\zeta(u)=x$ (функция, обратная к $\nu(x)$)

$\xi(x)$ — номер той связной компоненты П-комплекса X ,которая содержит вершину со знаком ω

$\tau(x)$ — суперпозиция ξ и ζ: $\tau(x)=\zeta(\xi(x))$.

§ 1. Вспомогательные определения, касающиеся комплексов.

Множество K (конечное или бесконечное) об'ектов двух родов - вершин и отрезков - мы назовем комплексом, если выполняются следующие условия:

1/ элементы K образуют одномерный топологический комплекс;

2/ на вершинах K задана характеристическая функция $f_K(e)$, принимающая целые положительные значения,

3/ каждой вершине e сопоставлено некоторое множество $Z^K(e)$ целых положительных чисел (значков); множество $Z^K(e)$ приведено во взаимно однозначное соответствие с множеством $O^K(e)$ отрезков, отходящих от вершины e.

Если множество K конечно, то комплекс назовем конечным; в противном случае назовем комплекс бесконечным. Пустое множество будем считать частным случаем конечного комплекса.

Наглядно можно представить себе, что каждый отрезок комплекса несет на себе два значка - по значку на каждом своем конце. Если взять некоторую вершину e и рассмотреть множество значков, стоящих на обращенных к этой вершине концах отрезков, отходящих от этой вершины, то это множество и есть $Z^K(e)$.

Множество L назовем подкомплексом комплекса K,

если выполняются следующие условия:

1/ L есть подкомплекс K в топологическом смысле;

2/ на вершинах L определена целочисленная функция $f_L(e)$ причем $f_L(e) = f_K(e)$;

3/ каждой вершине из L сопоставлено некоторое множество $Z^L(e)$ целых положительных чисел (значков); множество $Z^L(e)$ приведено во взаимно-однозначное соответствие с множеством $O^L(e)$ отрезков, отходящих от вершины e и принадлежащих L x/. Требуется, чтобы:

а/ $Z^L(e) \subseteq Z^K(e)$,

б/ соответствие между $Z^L(e)$ и $O^L(e)$ было тем, которое индуцируется соответствием между $Z^K(e)$ и $O^K(e)$.

Множество $\mathfrak{O}$ назовем подклассом комплекса K, если выполняются следующие условия:

1/ $\mathfrak{O}$ есть подкомплекс K в топологическом смысле;

2/ на вершинах $\mathfrak{O}$ определена целочисленная функция $f_{\mathfrak{O}}(e)$ причем $f_{\mathfrak{O}}(e) = f_K(e)$;

3/ каждой вершине e из $\mathfrak{O}$ сопоставлено некоторое множество $Z^{\mathfrak{O}}(e)$ целых положительных чисел(значков); множество $Z^{\mathfrak{O}}(e)$ приведено во взаимно-однозначное соответствие с множеством $O^K(e)$ отрезков, отходящих от вершины e и принадлежащих комплексу K. Требуется, чтобы

а/ $Z^{\mathfrak{O}}(e) = Z^K(e)$,

б/ соответствие между $Z^{\mathfrak{O}}(e)$ и $O^K(e)$ было тем

---

x/ При этом, очевидно $O^L(e) \subseteq O^K(e)$

же, что и между $Z^K(e)$ и $O^K(e)$.

Таким образом, подкомплекс L отличается от подкласса $\mathfrak{A}$ тем, что в случае подкомплекса L мы ассоциируем с каждой вершиной лишь те значки, которые принадлежат лишь отрезкам из L, инцидентным с этой вершиной, а в случае подкласса $\mathfrak{A}$ мы ассоциируем с каждой вершиной все отрезки из K, инцидентные с этой вершиной. Иными словами, из вершины подкомплекса мы можем "видеть" только те отходящие от нее отрезки, которые принадлежат L, а из вершины подкласса - все отходящие от нее отрезки.

Подкомплекс (подкласс) будем называть натянутым на множество своих вершин.

Две вершины назовем соседними, если они соединены отрезком. Вершина называется соседней с множеством A, если она не принадлежит A и является соседней хотя бы с одной вершиной из A. Вершина называется крайней для множества A, если она принадлежит A и является соседней хотя бы с одной вершиной, не принадлежащей A. Замыканием $[A]$ множества A называется минимальный подкомплекс, содержащий как множество A, так и множество всех вершин, соседних с A. Отрезки, соединяющие вершины из A с вершинами, не принадлежащими к A, называются внешними для A. Очевидно, внешние для A отрезки соединяют крайние вершины для A с вершинами, соседними с A.

Подкласс отличается от подкомплекса тем, что, имея

дело с подклассом, мы обращаем внимание на внешние отрезки.

Комплекс K назовем ограниченным, если ограничено множество $\{f_K(e)\}$ значений характеристической функции и ограничено множество $\bigcup_e Z^K(e)$ (т.-е. совокупное множество всех значков, встречающихся в K). Всюду в работе, не делая специальных оговорок, рассматриваются только ограниченные комплексы. Ограниченному комплексу припишем порядок (n, α), где

$$n = \sup\{f_K(e)\} \quad , \quad \alpha = \sup \bigcup_e Z^K(e).$$

В множестве порядков комплексов можно ввести частичное упорядочивание, положив

$$(n_1, \alpha_1) \leq (n_2, \alpha_2)$$

если одновременно $n_1 \leq n_2$ и $\alpha_1 \leq \alpha_2$.

§ 2. Запись некоторых алгоритмов в терминах алгоритма Колмогорова.

1.

Во Введении было отмечено, что алгоритмы в смысле других определений автоматически записываются в терминах алгоритма Колмогорова.

При изображении на бумаге П-комплексов условимся о следующем:

1/ активные вершины будем выделять, обводя их красными кружками;

2/ значки, стоящие на концах отрезков, будем для простоты опускать;

3/ разрешим характеристической функции $f_K^{(e)}$ принимать в качестве значений не только натуральные числа, но и некоторые символы (например, буквы из алфавитов А.А.Маркова). Если угодно, можно считать, что эти символы просто обозначают некоторые натуральные числа.

Для записи любого алгоритма в терминах алгоритма Колмогорова нужно расчленить записываемый алгоритм на отдельные, достаточно четкие операции.

2.

Покажем, как записать алгоритм Маркова в терми-

нах алгоритма Колмогорова. Мы предполагаем, что читатель знаком с определением нормального алгоритма по статье [1] .

Слово $\alpha_1 \alpha_2 \dots \alpha_n$ из какого-либо алфавита Маркова мы будем изображать комплексом

$$\overset{\alpha_1}{\circ} \!-\!\!-\!\!-\! \overset{\alpha_2}{\circ} \!-\!\!-\!\!-\! \dots \!-\!\!-\! \overset{\alpha_n}{\circ}$$

который для простоты будем записывать в виде

$$\alpha_1 \alpha_2 \dots \alpha_n$$

считая, что между соседними буквами проведены отрезки. В случае, если потенциальный подкомплекс несвязен, отдельные его компоненты будем отделять синей вертикальной чертой. Например, для П-комплекса

$$a\,b\,c\,d\,e\,f\,g\,h$$

потенциальный подкомплекс запишем так:

$$a\,b\,c \mid e\,f\,g\,h$$

Нормальный алгоритм задается схемой

$(\mathfrak{A}_1)$	$A_1 \to B_1$
$(\mathfrak{A}_2)$	$A_2 \to B_2$
....	....
$(\mathfrak{A}_r)$	$A_r \to B_r$

Пусть слово A_k имеет вид $a_1^{(k)} a_2^{(k)} \dots a_{m_k}^{(k)}$

а слово B_k - вид $b_1^{(k)} b_2^{(k)} \dots b_{n_k}^{(k)}$.

Для простоты будем считать, что все слова A_k, B_k не пусты.

Укажем, как задать соответствующий алгоритм Колмогорова.

Мы хотим применить нормальный алгорифм к слову

$$A = a_1 a_2 \dots a_q$$

которое мы запишем в виде П-комплекса $\tilde{A}$

$$\textcircled{a_1}\, a_2 \dots a_q$$

Расчленим процесс применения нормального алгорифма на отдельные операции.

Сперва мы припишем к П-комплексу $\tilde{A}$ слева цифру 1. Это значит, что мы будем применять к нему подстановку (α_1). В терминах алгоритма Колмогорова приписывание цифры 1 задается парой (H, H^*) из множества $\mathfrak{M}$ правил переработки (пару (H, H^*) нам будет удобнее писать в виде $H \to H^*$):

$$(I) \qquad \textcircled{a_1}\, a_2 \longrightarrow \textcircled{1}\, a_1 \; a_2$$

Пунктиром будем показывать то взаимно-однозначное соответствие, которое установлено между граничными вершинами H и некоторым множеством вершин H^*. Мы будем пунктир опускать там, где это соответствие очевидно (как, например, в (I)).

Чтобы алгорифм можно было применять к любому слову A из алфавита, надо чтобы в правиле (I) и последующих правилах переработки под буквами $a_1, a_2, \dots$ понимались <u>любые</u> буквы из рассматриваемого алфавита.

Поэтому правило (I) (как и другие правила) на самом деле является серией правил (если в алфавите ν букв, то (I) является серией из ν^2 правил).

Теперь мы должны применять подстановку (α_1). Для этого сперва выделим в A группу из m_1 букв (m_1 - число букв в A_1). Мы выделим эту группу в качестве потенциального подкомплекса. Это достигается последовательностью правил:

$$(\Pi^1) \quad \textcircled{1}\, a_1 \to \textcircled{1}\textcircled{a_1}$$

$$(\Pi^2) \quad \textcircled{1}\textcircled{a_1}\, a_2 \to \textcircled{1}\textcircled{a_1}\textcircled{a_2}$$

$$(\Pi^3) \quad \textcircled{1}\textcircled{a_1}\textcircled{a_2}\, a_3 \to \textcircled{1}\textcircled{a_1}\textcircled{a_2}\textcircled{a_3}$$

. . . .

$$(\Pi^{m_1-1}) \quad \textcircled{1}\textcircled{a_1}\textcircled{a_2}\dots\textcircled{a_{m_1-2}}\, a_{m_1-1} \to \textcircled{1}\textcircled{a_1}\textcircled{a_2}\dots\textcircled{a_{m_1-2}}\textcircled{a_{m_1-2}}$$

При этом может случиться, что в слове A меньше букв, чем в A_1 и мы дойдем до конца слова A , не успев выделить m_1 букв. Это значит, что A не содержит вхождений слова A_1 и мы должны начать применять (α_2) . Поэтому мы должны иметь следующую совокупность правил:

$$(\text{III}^p) \quad \textcircled{1}\textcircled{a_1}\textcircled{a_2}\dots\textcircled{a_p} \to \textcircled{2}\, a_1 a_2 \dots a_p \qquad (\text{для всех } p < m_1)$$

Появление цифры 2 и означает, что мы будем применять (α_2).

Если этого не случилось, то мы благополучно выделили в A потенциальное множество из m_1 первых (считая слева) букв и получили П-комплекс

$$\underbrace{\textcircled{1}\textcircled{a_1}\textcircled{a_2}\dots\textcircled{a_{m_1-1}}\, a_{m_1}}_{\text{потенциальная часть}}\, a_{m_1+1}\dots a_q$$

Теперь мы должны посмотреть, совпадает ли слово $a_1 a_2 \ldots a_{m_1-1} a_{m_1}$ со словом $A_1 = a_1^{(1)} a_2^{(1)} \ldots a_{m_1}^{(1)}$.

Пусть совпадает. Тогда его надо заменить на $B_1 = b_1^{(1)} b_2^{(1)} \ldots b_{n_1}^{(1)}$.

Таким образом, должно быть правило

$$(\mathrm{IV})\quad 1\, a_1^{(1)} a_2^{(1)} \ldots a_{m_1-1}^{(1)} a_{m_1}^{(1)} \to 1\, b_1^{(1)} \ldots b_{n_1-1}^{(1)} b_{n_1}^{(1)}$$

Появление цифры 1 говорит о том, что к получившемуся комплексу надо снова применять подстановку (α_1).

Если $A_1 \to B_1$ заключительная подстановка, что записывается $A_1 \to . B_1$, то правило (IV) видоизменится так

$$(\mathrm{V}^1)\quad 1\, a_1^{(1)} \ldots a_{m_1-1}^{(1)} a_{m_1}^{(1)} \to \omega\, b_1^{(1)} \ldots b_{n_1-1}^{(1)} b_{n_1}^{(1)}$$

(V^2) $\omega a \to$ „стоп" (буква a – любая буква алфавита)

Пусть теперь слово $a_1 a_2 \ldots a_{m_1-1} a_{m_1}$ не совпадает с A_1. Тогда мы должны всю эту группу из m_1 букв передвинуть на одну букву направо:

$$(\mathrm{VI}^1)\quad 1\, a_1 a_2 \ldots a_{m_1-1} a_{m_1} \to 1\, a_1 a_2 \ldots a_{m_1-1} a_{m_1}$$

Это правило пишется для всех групп из m_1 букв $a_1 a_2 \ldots a_{m_1}$ отличных от A_1.

Применив (VI^1), мы получим П-комплекс

$$1\, a_1 a_2 a_3 \ldots a_{m_1} a_{m_1+1} \ldots a_q$$

в котором надо исследовать на предмет совпадения с A_1 группу из m_1 букв $a_2 a_3 \ldots a_{m_1} a_{m_1+1}$

Если эта группа совпадает с A_1, мы заменяем ее на B_1, если нет – сдвигаем направо еще на одну букву.

Если слово A вообще не содержит вхождений слова A_1 то мы пойдем до конца этого слова и перейдем к применению подстановки $(\mathfrak{N}_2)$. Это осуществляется правилом

$$(VII) \quad 1\, a_1 \mid a_s\, a_{s+1} \cdots a_{s+m_1-1} \longrightarrow 2\, a_1 \mid a_s\, a_{s+1} \cdots a_{s+m_1-1}$$

И так далее. Каждый раз мы применяем одну из подстановок $(\mathfrak{N}_k)$ смотря по тому, какая цифра стоит спереди. Если слово A не содержит вхождений слова A_k , то мы перейдем к $(\mathfrak{N}_{k+1})$, так же как от $(\mathfrak{N}_1)$ мы перешли к $(\mathfrak{N}_2)$.

Процесс длится до тех пор, пока

или 1/ произойдет заключительная подстановка

или 2/ мы просмотрим все подстановки $(\mathfrak{N}_1), (\mathfrak{N}_2), \ldots, (\mathfrak{N}_r)$ и не обнаружим в A вхождений ни A_1, ни A_2, ..., ни A_r.

В обоих этих случаях цифра, стоящая спереди, "гаснет", вместо нее "вспыхивает" знак ω , после чего работает уже написанное правило

$$(V^2) \qquad \omega\, a \longrightarrow \text{„стоп"}$$

Все остальные, не выписанные нами правила совершенно аналогичны, их нетрудно выписать в явном виде. Мы видим, что машина Колмогорова, осуществляющая нормальный алгорифм Маркова, строится хотя и утомительно, но совершенно автоматически.

3.

Так же автоматически записывается в терминах

алгоритма Колгоморова и вычисление произвольной частично-рекурсивной функции.

~~Именно~~, Назовём изображением числа **x** П-комплекс $\overline{X}$ вида.

$x+2$ раз

Изображением "энки" чисел $(x_1,\dots,x_n)$ назовем П-комплекс $\overline{(X_1,\dots,X_n)}$:

x_1+2 раз, x_2+2 раз, x_n+2 раз

который будем записывать также так:

$$\overline{X}_1-\overline{X}_2-\dots-\overline{X}_n$$

! Назовем функцию $y=f(x_1,\dots,x_n)$ **вычислимой**, если существует машина Колмогорова $\mathcal{T}_f$, перерабатывающая П-комплекс $\overline{(X_1,\dots,X_n)}$ в П-комплекс $\overline{Y}$

Мы хотим показать, что всякая частично-рекурисивная функция вычислима.

Частично-рекурсивная функция - сокращено чрф- определяется так [3], [7]. Задается серия начальных функций:

$$I_{nk}(x_1,\dots,x_n)=x_k,\quad O_n(x_1,\dots,x_n)=0,\quad S(x)=x+1,$$

и указываются рекурсивные операции, позволяющие по уже построенным функциям строить новые:

1/ /Суперпозиция/. Если $\varphi_1(x_1,\dots,x_n),\dots,\varphi_m(x_1,\dots,x_n)$; $\psi(x_1,\dots,x_m)$ - чрф, то

$$\rho(x_1,\dots,x_n)=\psi(\varphi_1(x_1,\dots,x_n),\dots,\varphi_m(x_1,\dots,x_n))$$

тоже чрф.

2/ /Примитивная рекурсия/. Если $h(x_1, \ldots, x_{n-1})$ и $g(x_1, \ldots, x_{n-1}, x_n, x_{n+1})$ – чрф

то и $f(x_1, \ldots, x_{n-1}, x_n)$ – чрф, где $f(x_1, \ldots, x_n)$ задается равенствами

$$f(x_1, \ldots, x_{n-1}, 0) = h(x_1, \ldots, x_{n-1})$$

$$f(x_1, \ldots, x_{n-1}, x_n+1) = g(x_1, \ldots, x_{n-1}, x_n, f(x_1, \ldots, x_n))$$

3/ (Применение Клиниевского оператора μ). Если $\Theta(x_1, \ldots, x_n; y)$ - чрф, то чрф и

$$\mu(x_1, \ldots, x_n) = \mu y[\Theta(x_1, \ldots, x_n; y) = 0]$$

Класс частично-рекурсивных функций определяется как минимальный класс, содержащий начальные функции и замкнутый относительно рекурсивных операций.

Чтобы доказать вычислимость любой частично-рекурсивной функции, достаточно доказать поэтому, что начальные функции вычислимы и что рекурсивные операции сохраняют вычислимость. Иными словами, надо во-первых построить машины Колмогорова для начальных функций, и во-вторых показать, как, имея машины для заданных функций, построить машину для функции, получающейся из заданных рекурсивной операцией.

Доказательство, таким образом, идет по индукции. При этом нам выгодно усилить индуктивное предположение. Именно, будем строить для чрф не просто машину, а <u>допустимую</u> машину.

Будем считать, что каждой частично-рекрусивной функции отнесен присущий ей одной функциональный

знак f, φ, ψ и т.д. /, причём в этом функциональном знаке содержится информация о всех рекурсивных операциях, создавших эту функцию. Мы скажем, что для функции $y = f(x_1, \ldots, x_n)$ построена допустимая машина T_f, если T_f перерабатывает любой комплекс вида

$$K_1 \; \overset{\alpha}{o} - \overset{,}{o} - \overset{f}{o} - \overset{(}{o} - \overset{,}{o} - \underbrace{\overset{2}{o} \ldots \overset{2}{o}}_{x_1+2 \text{ раз}} - \overset{,}{o} \ldots - \overset{,}{o} - \underbrace{\overset{2}{o} \ldots \overset{2}{o}}_{x_n+2 \text{ раз}} - \overset{,}{o} - \overset{)}{o} - \overset{,}{o} - \overset{\beta}{o} \; K_2$$

/ где K_1 и K_2 -произвольные комплексы, α один из знаков: ((левая скобка) или , (запятая), а β один из знаков) или , в комплексе вида

$$K_1 \; \overset{\alpha}{o} - \overset{,}{o} - \underbrace{\overset{2}{o} - \overset{2}{o} \ldots \overset{2}{o}}_{y+2 \text{ раз}} - \overset{,}{o} - \overset{\beta}{o} \; K_2$$

(с теми же самыми K_1, K_2, α, β) /. Допустимость машины означает попросту говоря, что в процессе переработки мы не выйдем за пределы комплекса

$$\overset{\alpha}{o} - \overset{,}{o} - \overset{f}{o} - \overset{(}{o} - \overset{,}{o} - \overset{2}{o} \ldots \overset{2}{o} - \overset{,}{o} \ldots \overset{,}{o} - \overset{2}{o} \ldots \overset{2}{o} - \overset{,}{o} - \overset{)}{o} - \overset{,}{o} - \overset{\beta}{o}$$

Все машины, которые будут строиться в этом паррграфе, допустимы.

(2,01) Для простоты комплекс вида

$$\overset{,}{o} - \overset{f}{o} - \overset{(}{o} - \overset{,}{o} - \overset{2}{o} \ldots \overset{2}{o} - \overset{,}{o} \ldots \overset{,}{o} - \overset{2}{o} \ldots \overset{2}{o} - \overset{,}{o} - \overset{)}{o} - \overset{,}{o}$$

(2,02) будем записывать

$$,f(,2\ldots2,\ldots,2\ldots2,),$$

считая соседние символы связанными отрезком. Если же они не связаны, то, так же как и раньше, условимся ставить между ними вертикальную синюю черту.

Еще короче комплекс мы будем записывать так:

(2,03)

$$,f(\overline{x}_1 - \overline{x}_2 - \ldots - \overline{x}_n),$$

Итак, сперва надо построить допустимые машины для начальных функций. Покажем это на примере функции $S(x)=x+1$

Задается П- комплекс

$$,S(,\underbrace{22\ldots2}_{x+2\ \text{раз}},)$$

Надо его переработать в П-комплекс

$$,\underbrace{22\ldots2}_{x+3\ \text{раз}},$$

Это достигается правилами переработки

$,S(\longrightarrow ,SC$

$,SC, \longrightarrow ,SC,$

$,SC,2 \longrightarrow ,SC,22$ (здесь возникает тот самый $(x+3)$-й символ 2)

$,SC \mid 222 \longrightarrow ,SC \mid 222$

$,SC \mid 22, \longrightarrow ,SC \mid 22,$

$,SC, \mid 2,) \longrightarrow ,SC, \mid 2,)$

$,SC, \mid ,), \longrightarrow ,SC, \mid ,),$

$,SC, \mid 2,), \longrightarrow ,SC, \mid 2,),$

$,SC,2 \mid 2,), \longrightarrow ,2 \mid 2,$

~~Для всех групп вида~~

Теперь надо показать, что если для некоторых функций существуют допустимые машины, то и для функции, полученной из них рекурсивной операцией, существует допустимая машина.

Суперпозиция. Пусть для функций $\varphi_1(x_1,\dots,x_n),\dots \varphi_m(x_1,\dots,x_n), \psi(x_1,\dots,x_m)$ построены допустимые машины $\mathcal{T}_{\varphi_1},\dots,\mathcal{T}_{\varphi_m},\mathcal{T}_\psi$. Нужно построить допустимую машину $\mathcal{T}_\rho$ для функции

$$\rho(x_1,\dots,x_n)=\psi(\varphi_1(x_1,\dots,x_n),\dots,\varphi_m(x_1,\dots,x_n))$$

Ее построить легко. Действительно, легко построить допустимую машину со следующими свойствами.

Если дан П-комплекс

(2.04) $$,\rho(\bar{X}_1-\bar{X}_2-\dots-\bar{X}_n),$$

то она перерабатывает его в П-комплекс

(2.05) $$,\psi(,\varphi_1(\bar{X}_1-\bar{X}_2-\dots-\bar{X}_n),,\varphi_2(\bar{X}_1-\dots-\bar{X}_n),\dots,\varphi_m(\bar{X}_1-\dots-\bar{X}_n),),$$

После этого пускается в ход машина $\mathcal{T}_{\varphi_1}$, которая заменяет в (2,05) подкомплекс $,\varphi_1(\bar{X}_1-\dots-\bar{X}_n),$ на $\bar{Z}_1$ - ($\bar{Z}_1$ изображение числа z_1, где $z_1=\varphi_1(x_1,\dots x_n)$). Получаем П-комплекс

(2,06) $$,\psi(,\underbrace{22\dots2}_{z_1\ \text{раз}},,\varphi_2(\bar{X}_1-\dots-\bar{X}_n),\dots,\varphi_m(\bar{X}_1-\dots-\bar{X}_n),),$$

Легко написать правила, перерабатывающие его в

(2,07) $$,\psi(\bar{Z}_1,\varphi_2(\bar{X}_1-\dots-\bar{X}_n),\dots,\varphi_m(\bar{X}_1-\dots-\bar{X}_n),),$$

А тогда пускается в ход $\mathcal{T}_{\varphi_2}$ и получается

(2,8) $$,\psi(\bar{Z}_1,\underbrace{22\dots2}_{z_2\ \text{раз}},,\varphi_3(\bar{X}_1-\dots-\bar{X}_n),\dots,\varphi_m(\bar{X}_1-\dots-X_n),),$$

где

$$z_2=\varphi_2(x_1,\dots,x_n)$$

 16

И так далее. В конце концов получим

(2,09) $,\varphi(\overline{Z}_1 - \overline{Z}_2 - \ldots - \overline{Z}_{m-1}, \underbrace{22\ldots2}_{z_m \text{ раз, где } z_m = \varphi_m(x_1,\ldots,x_n)},)$,

Этот комплекс нетрудно переработать в комплекс

(2,10) $,\varphi(\overline{Z}_1 - \overline{Z}_2 - \ldots - \overline{Z}_{m-1})$,

После чего пускается в ход машина T_φ, которая перерабатывает (2,10) в $\overline{Y}$. ($\overline{Y}$ изображает y, где $y = \rho(x_2,\ldots,x_n)$).

<u>Примитивная рекурсия.</u> Для простоты рассмотрим случай функции от одного аргумента, задаваемой рекурсивными равенствами

$$f(0) = h$$
$$f(x) = g(x-1, f(x-1)).$$

Легко построить машину T_f, которая перерабатывает комплекс

(2,11) $,f(\overline{X})$,

1/ если $x=0$ - в $\overline{H}$ (изображение h)

2/ если $x>0$ - в комплекс

(2,12) $,g(\overline{X-1}, f(\overline{X-1}),)$,

где $\overline{X-1}$ - изображение числа $x-1$ ← знак связи („отрезок")

Так же машина, работая дальше, переработает комплекс (2,12)

2. 1/ если $x-1=0$ - в комплекс

$,g(\overline{X-1}, \underbrace{22\ldots2}_{h+2 \text{ раза}},)$,

2. 2/ если $x-1>0$ - в комплекс

(2,13) $,g(\overline{X-1}, g(\overline{X-2}, f(\overline{X-2}),),)$,

И так далее. В конце концов, комплекс (2,11) переработается в комплекс

17

$$(2,14) \quad , g(\overline{x-1}, g(\overline{x-2}, g(\overline{x-3}, \ldots, g(\overline{x-k}, \underbrace{2\,2 \ldots 2}_{h+2 \text{ раз}}),),), \ldots),$$

Нетрудно теперь задать правила переработки, которые бы превращали комплекс (2,14) в изображение числа $y = f(x)$. Для этого комплекс (2,14) надо рассмотреть как формулу и произвести ее свертывание, начиная с внутренних скобок.

Оператор μ. Несколько более громоздко, но тоже, если подумать, достаточно просто, дается приём, как, зная машину $\mathfrak{T}_\theta$ для $\theta(x_1, \ldots, x_n; y)$ построить машину $\mathfrak{T}_\mu$ для

$$\mu(x_1, \ldots, x_n) = \mu y\left[\theta(x_1, \ldots, x_n; y) = 0\right]$$

Для этой машины нужно написать 31 правило переработки (не считая правил, задающих $\mathfrak{T}_\theta$, которые тоже включаются в задание $\mathfrak{T}_\mu$). Вероятно, это число можно сократить. Мы не будем выписывать эти правила.

Таким образом, оказывается, что для всякой частично-рекурсивной функции $y = f(x_1, \ldots, x_n)$ существует допустимая машина, перерабатывающая П-комплекс

$$, f(\overline{x_1} - \overline{x_2} - \ldots - \overline{x_n})$$

в П"комплекс $\overline{y}$

Отсюда следует, что все частично-рекурсивные функции вычислимы.

§3. Рекурсивность алгоритма Колмогорова.

1.

В этом параграфе показывается рекурсивность безусловного алгоритма Колмогорова.

Для этого устроим "арифметизацию" этого алгоритма. Каждому П-комплексу K мы отнесем натуральное число k - его номер.

Произведем сперва одно усовершенствование машины Колмогорова. Именно, исключим из правил переработки слово "стоп". Будем считать, что конец процесса наступает тогда, когда на одной (и единственной) из вершин П-комплекса появляется символ ω. Связная компонента этой вершины и есть решение. Вместо пары $(H,$ «стоп»$)$ будем, таким образом, писать пару (H, H^{ω}), где H^{ω} отличается от H только тем, что на единственной (по условию) активной вершине H поставлен символ ω. Ясно, что для каждой машины Колмогорова в старом смысле можно построить машину в новом смысле (и обратно), задающую ту же функцию от комплексов (с точностью до символа ω на активной вершине).

Соответственно расширим понятие П-комплекса, включив в число значений характеристической функции символ ω. Впрочем, этот символ можно считать просто некоторым фиксированным натуральным числом. Отождествим его,

например, с единицей.

2.

Между П-комплексом и его номером стоит промежуточный об'ект-таблица. Занумеруем вершины П-комплекса K в каком-нибудь порядке: $e_1, e_2, \ldots, e_v$. Составим таблицу T_K:

0	α_{01}	α_{02}	$\ldots$	α_{0v}
α_{10}	α_{11}	α_{12}	$\ldots$	α_{1v}
α_{20}	α_{21}	α_{22}	$\ldots$	α_{2v}
$\ldots$	$\ldots$	$.$	$\ldots$	$.$
$.$	$.$	$.$	$\ldots$	$.$
$.$	$.$	$.$	$\ldots$	$.$
α_{v0}	α_{v1}	α_{v2}	$\ldots$	α_{vv}

Мы полагаем $\alpha_{ij} = 0$, если e_i и e_j не соединены отрезком. Если же они соединены отрезком, то полагаем α_{ij} равным тому значку, который стоит в конце этого отрезка, обращенном к e_i. Кроме того, положим $\alpha_{00} = 0,\ \alpha_{0j} = \alpha_{j0} = 2^{\chi(e_j)} 3^{f_K(e_j)}$

Здесь $\chi(e)$ – функция, выделяющая активные

вершины: $\chi(e) = 1$, если e активна, и $\chi(e) = 0$ в противном случае; $f_{\alpha}(e)$ - характеристическая функция, о которой шла речь выше (см. стр)

Две таблицы, отвечающие одному и тому же П-комплексу (и соответствующие разным нумерациям вершин этого П-комплекса) назовем <u>эквивалентными.</u> Каждому П-комплексу с числом вершин v отвечает множество из $v!$ эквивалентных таблиц (столько, сколько различных нумераций вершин П-комплекса); некоторые из этих таблиц могут совпадать.

Введем порядок в множестве таблиц. Именно, запишем таблицу в строчку:

$$\alpha_{00}\alpha_{01}\ldots\alpha_{0v}\alpha_{10}\alpha_{11}\ldots\alpha_{1v}\ldots\alpha_{k0}\alpha_{k1}\ldots\alpha_{kv}\ldots\alpha_{v0}\alpha_{v1}\ldots\alpha_{vv}$$

А среди строчек введем словарный порядок.

Назовем таблицу нормальной, если нумерация вершин П"комплекса устроена так, что сперва идут все вершины из множества A активных вершин, затем все вершины из множества F граничных вершин, а затем - вершины из множества G пассивных вершин. (см. стр. 13).

Нормальная таблица имеет вид

		A			F			G		
	0	α_{01}	$\cdots$	α_{0a}	α_{0a+1}	$\cdots$	α_{0h}	α_{0h+1}	$\cdots$	α_{0v}
A	α_{10}	α_{11}	$\cdots$	α_{1a}	α_{1a+1}	$\cdots$	α_{1h}	α_{1h+1}	$\cdots$	α_{1v}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	α_{a0}	α_{a1}	$\cdots$	α_{aa}	α_{aa+1}	$\cdots$	α_{ah}	α_{ah+1}	$\cdots$	α_{av}
F	α_{a+10}	α_{a+11}	$\cdots$	α_{a+1a}	α_{a+1a+1}	$\cdots$	α_{a+1h}	α_{a+1h+1}	$\cdots$	α_{a+1v}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	α_{h0}	α_{h1}	$\cdots$	α_{ha}	α_{ha+1}	$\cdots$	α_{hh}	α_{hh+1}	$\cdots$	α_{hv}
G	α_{h+10}	α_{h+11}	$\cdots$	α_{h+1a}	α_{h+1a+1}	$\cdots$	α_{h+1h}	α_{h+1h+1}	$\cdots$	α_{h+1v}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	α_{v0}	α_{v1}	$\cdots$	α_{va}	α_{va+1}	$\cdots$	α_{vh}	α_{vh+1}	$\cdots$	α_{vv}

(H = A ∪ F for both rows and columns.)

Мы будем употреблять для нормальной таблицы такую запись:

	A	F	G
A	AA	AF	AG
F	FA	FF	FG
G	GA	GF	GG

Заметим, что потенциальному подкомплексу H отвечает нормальная таблица T_H, занимающая левый верхний угол нормальной таблицы T_K:

	A	F
A	AA	AF
F	FA	FF

и что всегда

$$AG = GA = \begin{vmatrix} 0\,0 \ldots 0 \\ 0\,0 \ldots 0 \\ \cdots\cdots \\ 0\,0 \ldots 0 \end{vmatrix}$$

Рассмотрим множество $\{T_K\}$ нормальный таблиц, отвечающих П-комплексу K. В левом верхнем углу T_K помещается таблица T_H отвечающая П-комплексу H. Отберем те T_K, у которых T_H минимальны (в смысле нашего порядка). Получим множество $\{\tilde{T}_K\}$. Из множества $\{\tilde{T}_K\}$ выберем теперь минимальную таблицу, которую назовем каноничеcкой таблицей для П-комплекса K. Каждому П-комплексу эффективно и однозначно соответствует каноническая таблица и обратно по каждой канонической таблице эффективно и однозначно восстанавливается П-комплекс.

Весь алгоритм переписывается на языке канонических таблиц.

Состояние машины есть каноническая числовая таблица T. Каноничность таблицы усматривается непосредственно из ее вида. В самом деле, легко обнаружить, отвечает таблица некоторому П-комплексу или нет (чтобы таблица отвечала, некоторому П-комплексу необходимо и достаточно, чтобы $\alpha_{oj} = \alpha_{jo}$, все α_{oj} имели вид 3^n или 2.3^n и, наконец, нули в таблице были расположены симметрично). Если же таблица отвечает некоторому П-комплексу, то в ней легко обнаружить активные столбцы и строки (т.-е. столбцы с тем номером j, для которых α_{oj} имеет вид 2.3^n и строки с тем номером j, для

которых α_{j0} имеет вид 2.3^n). Столбец (строка) называется потенциальным, если он активный или если он пересекается с некоторой активной строкой (столбцом) по отличному от нуля элементу. Элементы, стоящие на пересечении потенциальных столбцов и строк, образуют потенциальную таблицу T_H . Таблица T называется канонической, если ее потенциальная таблица занимает в ней левый верхний угол.

Итак, состояние машины есть каноническая таблица T; за один шаг машина, находящаяся в состоянии T , или

1/ перерабатывает T в T' ,

2/ дает сигнал о конце процесса (в том случае, когда одно из чисел α_{0j} имеет вид 2.3^ω x/.)

или 3/ останавливается безрезультатно.

Переработка T в T' происходит на основании правил переработки, которые мы запишем так:

Столбиком выпишем потенциальные таблицы $T_{H_1}, T_{H_2}, \ldots, T_{H_s}$. (они отвечают первым элементам в парах (H, H^*)). Против каждой из них выписывается таблица T_{H^*} . Получается такая схема:

T_{H_1}	$T_{H_1^*}$
T_{H_2}	$T_{H_2^*}$
.	. . . . .
T_{H_s}	$T_{H_s^*}$

---

x/ А так как мы отождествили ~~ω~~ ω с единицей, то это число попросту равно 6.

Каждая таблица T_{H_r} - каноническая. Таблица $T_{H_r^*}$ не каноническая (и даже не нормальная), а вот какая. Таблица T_{H_r} имеет вид

		A			F		
	0	α_{01}	$\dots$	α_{0a}	$\alpha_{0\,a+1}$	$\dots$	$\alpha_{0\,a+m}$
A	α_{10}	α_{11}	$\dots$	α_{1a}	$\alpha_{1\,a+1}$	$\dots$	$\alpha_{1\,a+m}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	α_{a0}	α_{a1}	$\dots$	α_{aa}	$\alpha_{a\,a+1}$	$\dots$	$\alpha_{a\,a+m}$
F	$\alpha_{a+1\,0}$	$\alpha_{a+1\,1}$	$\dots$	$\alpha_{a+1\,a}$	$\alpha_{a+1\,a+1}$	$\dots$	$\alpha_{a+1\,a+m}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$\alpha_{a+m\,0}$	$\alpha_{a+m\,1}$	$\dots$	$\alpha_{a+m\,a}$	$\alpha_{a+m\,a+1}$	$\dots$	$\alpha_{a+m\,a+m}$

=

	A	F
A	AA	AF
F	FA	FF

где нумерация $e_1, \dots, e_{a+m}$ устроена так, чтобы таблица была канонической. Тогда таблица $T_{H_r^*}$ имеет вид

		A^*			F^*		
	0	α^{1}_{01}	$\dots$	α^{1}_{0p}	$\alpha^{1}_{0\,p+1}$	$\dots$	$\alpha^{1}_{0\,p+m}$
A^*	α^{1}_{10}	α^{1}_{11}	$\dots$	α^{1}_{1p}	$\alpha^{1}_{1\,p+1}$	$\dots$	$\alpha^{1}_{1\,p+m}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	α^{1}_{p0}	α^{1}_{p1}	$\dots$	α^{1}_{pp}	$\alpha^{1}_{p\,p+1}$	$\dots$	$\alpha^{1}_{p\,p+m}$
F^*	$\alpha^{1}_{p+1\,0}$	$\alpha^{1}_{p+1\,1}$	$\dots$	$\alpha^{1}_{p+1\,p}$	$\alpha^{1}_{p+1\,p+1}$	$\dots$	$\alpha^{1}_{p+1\,p+m}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$\alpha^{1}_{p+m\,0}$	$\alpha^{1}_{p+m\,1}$	$\dots$	$\alpha^{1}_{p+m\,p}$	$\alpha^{1}_{p+m\,p+1}$	$\dots$	$\alpha^{1}_{p+m\,p+m}$

=

	A^*	F^*
A^*	A^*A^*	A^*F^*
F^*	F^*A^*	F^*F^*

Здесь $\alpha'_{oj} = \alpha'_{jo} = 2^{\chi'(e_j)} 3^{f'_k(e_j)}$

где $f'_k(e)$ - новая характеристическая функция, а $\chi'(e)$ - новое распределение активных вершин. При этом нумерация $e'_1, \dots, e'_{p+m}$ произвольна (так что A^* не есть, вообще говоря, активное множество), с тем только условием, чтобы вершины $e'_{p+1}, \dots, e'_{p+m}$ совпадали с вершинами $e_{a+1}, \dots, e_{a+m}$ (ведь при переработке H в H^* вершины из F не меняются). Отрезки, соединяющие вершины $e_{a+1}, \dots\dots, e_{a+m}$ также остаются неизменными (хотя значки на них могут и измениться), поэтому в таблицах

$$\begin{vmatrix} \alpha_{a+1\,a+1} & \dots & \alpha_{a+1\,a+m} \\ \vdots & \dots & \vdots \\ \alpha_{a+m\,a+1} & \dots & \alpha_{a+m\,a+m} \end{vmatrix} \quad \text{и} \quad \begin{vmatrix} \alpha_{p+1\,p+1} & \dots & \alpha_{p+1\,p+m} \\ \vdots & \dots & \vdots \\ \alpha_{p+m\,p+1} & \dots & \alpha_{p+m\,p+m} \end{vmatrix}$$

нули расположены на одинаковых местах.

Итак, таблица

	A	F
A	AA	AF
F	FA	FF

~~перерабатыва~~ заменяется ~~этой в~~ на

	A*	F*
A*	A*A*	A*F*
F*	F*A*	F*F*

Вершины F совпадают с вершинами F^* и в квадратах FF и F^*F^* нули расположены на одинаковых местах.

Пусть теперь дана каноническая таблица T, в

левом верхнем углу которой стоит потенциальная таблица T_H :

$T =$

	A	F	G
A	AA	AF	0
F	FA	FF	FG
G	0	GF	GG

$T_H =$

	A	F
A	AA	AF
F	FA	FF

Смотрим, находится ли T_H в левом столбце схемы переработки, и если нет, останавливаемся безрезультатно, а если да, то заменяем T_H на T_{H^*} и вставляем T_{H^*} вместо T_H в таблицу T :

	A^*	F^*	G^*
A^*	A^*A^*	A^*F^*	0
F^*	F^*A^*	F^*F^*	F^*G^*
G^*	0	G^*F^*	G^*G^*

При этом $G^* = G$ и участок

	FG
GF	GG

не меняется:

	FG
GF	GG

=

	F^*G^*
G^*F^*	G^*G^*

После этого мы строим каноническую таблицу T' , эквивалентную T^* (ясно, что T' строится по T^* эффективно). Таблица T' и есть результат одного шага переработки таблицы T .

Если T такова, что она дает сигнал о конце процесса, то нужно построить таблицу T_ξ , отвечающую свя-

ной компоненте вершины ω в том комплексе, которому отвечает таблица T. Таблица T_ξ - "связная компонента" T - строится по T эффективно; ее построение описывается аналогично тому, как мы описали построение T' по T.

3.

Перейдем теперь к нумерации алгоритма. Для этого запишем таблицу T в строчку

$$\alpha_{00}\ldots\alpha_{0v}\alpha_{10}\ldots\alpha_{1v}\ldots\alpha_{v0}\ldots\alpha_{vv}$$

Всего в строчке $(v+1)^2$ знаков. Сопоставим ей по Гёделю число

$$№(T) = p_0^{\alpha_{00}+1}\; p_1^{\alpha_{01}+1} \ldots p_v^{\alpha_{0v}+1} \ldots p_{(v+1)^2-1}^{\alpha_{vv}+1}$$

где p_i - i-тое простое число. Тогда каждой таблице отвечает число - ее номер.

Номером П-комплекса будем считать номер соответствующей ему канонической таблицы.

Рассмотрим теперь алгоритмическую функцию от комплексов

$$L = \Gamma(K)$$

Ей соответствует некоторая функция от канонических таблиц $T_L = \Gamma(T_K)$, которая, в свою очередь индуцирует функцию от номеров этих таблиц, или, как мы условились считать, номеров этих комплексов:

$$l = \gamma(k)$$

Покажем, что $\gamma(k)$ частично-рекурсивная функция.

Сделаем это так. Рассмотрим функцию $\sigma(k)$, такую, что если $k = №(T)$ то $\sigma(k) = №(T')$. Из того, как мы описали переработку T в T', ясно, что $\sigma(k)$ – примитивно-рекурсивная функция.

Введем функцию $\rho(k,m) = \underbrace{\sigma(\sigma \ldots \sigma(\sigma}_{m \text{ раз}}(k)) \ldots)$

Это есть номер канонической таблицы T^m, полученной на m-ом шаге переработки T. Функция $\rho(k,m)$ примитивно-рекурсивна, ибо

$$\rho(k,0) = k$$

$$\rho(k,m+1) = \sigma(\rho(k,m))$$

Итак, идет процесс. Получается последовательность таблиц

$T \quad T^1 \quad T^2 \ldots \quad T^m \ldots$

их номера $\rho(k,0)=k \quad \rho(k,1) \quad \rho(k,2) \ldots \quad \rho(k,m) \ldots$

Процесс продолжается до тех пор, пока на активной вершине T^m не возникнет ω, т.е. пока номер s таблицы T^m не будет удовлетворять равенству $\omega(s)=0$ (см. обозначения, стр. 31) Итак, процесс продолжается до первого m_0, удовлетворяющего равенству $\omega(\rho(k,m_0))=0$, т.-е. до $m_0 = \mu m[\omega(\rho(k,m))=0]$. Тогда $\rho(k,m_0)$ есть номер таблицы T^{m_0}, на которой процесс обрывается. Чтобы получить решение, надо построить "связную компоненту" T^{m_0}; ее номер есть, очевидно, примитивно-рекурсивная функция ξ от номера T^{m_0} (см. Обозначения). Итак, номер решения есть

$$\gamma(k) = \xi(\rho(k, \mu m[\omega(\rho(x,m))=0]))$$

т.е. частично-рекурсивная функция.

4.

В предыдущем параграфе было показано, что каждая частично-рекурсивная функция вычислима. Покажем теперь, что каждая вычислимая функция частично-рекурсивна.

Пусть дана вычислимая функция $y = f(x_1, \dots x_n)$ ~~Тогда~~ (По определению) существует алгоритм, перерабатывающий П-комплекс K, равный

$$\overline{X}_1 - \overline{X}_2 - \dots - \overline{X}_n$$

в П-комплекс L, равный $\overline{y}$. Как мы только что доказали, номер l комплекса L есть частично-рекурсивная функция от номера k комплекса K :

$$l = \gamma(k) .$$

С другой стороны, очевидно, что номер k есть примитивно-рекурсивная функция от чисел $x_1, \dots, x_n$:

$$k = \nu(x_1, \dots, x_n)$$

Также очевидно, что число y есть примитивно-рекурсивная функция от номера своего изображения

$$y = \zeta(l)$$

Итак,

$$f(x_1, \dots, x_n) = y = \zeta(\gamma(\nu(x_1, \dots, x_n)))$$

Следовательно, $f(x_1, \dots, x_n)$ - частично-рекурсивная функция.

Более того, было показано, что

$$\gamma(k) = \xi(\rho(k, \mu m[\omega(\rho(k, m)) = 0]))$$

где ξ и ω - некоторые вполне определенные примитивно-рекурсивные функции.

Таким образом

$$f(x_1,\dots,x_n)=\zeta(\xi(\rho(\nu(x_1,\dots,x_n),\mu m[\omega(\rho(\nu(x_1,\dots,x_n),m))=0])))$$

Если положить $\zeta(\xi(u))=\tau(u)$;

$$\rho(\nu(x_1,\dots,x_n),m)=g(x_1,\dots,x_n,m);$$

то получим окончательно

$$f(x_1,\dots,x_n)=\tau(g(x_1,\dots,x_n,\mu m[\omega(g(x_1,\dots,x_n,m))=0]))$$

При $n=1$ эта формула дает

$$f(x)=\tau(g(x,\mu m[\omega(g(x,m))=0]))$$

Мы получили тем самым новое каноническое выражение для произвольной частично-рекурсивной функции. Здесь τ и ω фиксированные, а g - произвольная примитивно-рекурсивные функции.

§ 4. Алгоритмическая сводимость.

1.

Прежде всего, уточним определение Тьюринговской сводимости, одновременно обобщив его на случай сводимости функций. Будем считать что функция $\gamma(x)$ сводится по Тьюрингу к функции $\delta(x)$, если осуществляется следующая конструкция. Машина, вычисляющая $\gamma(x)$, вырабатывает число m_1 и ставит вопрос «$\delta(m_1)=?$». В зависимости от значения $\delta(m_1)$ вырабатывается m_2 и ставится вопрос «$\delta(m_2)=?$». И так далее. Если все время давать ответы на вопросы «$\delta(m_i)=?$», то, наконец, мы придем к ответу и на вопрос «$\gamma(x)=?$»

Изложим все это более строгим языком. Каждое m_k есть вычислимая функция от всей предшествующей строчки.

$$x, m_1, \delta(m_1), m_2, \delta(m_2), \dots, m_{k-1}, \delta(m_{k-1})$$

Чтобы избежать функций с бесконечно-возрастающим числом аргументов, последовательность

$$\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_s$$

будем задавать ее гёделевским номером

$$e(\alpha_0, \alpha_1, \dots, \alpha_s) = p_0^{\alpha_0+1} p_1^{\alpha_1+1} \dots p_s^{\alpha_s+1}$$

где p_i — i-тое простое число.

Тогда

$$m_k = \psi(e(x, m_1, \delta(m_1), \dots, m_{k-1}, \delta(m_{k-1})))$$

где $\psi(u)$ — вычислимая, т.-е. частично-рекурсивная функция. Эту функцию ψ назовем сводящей.

Процесс построения чисел m_k продолжается до тех пор, пока мы не получим сигнал, что пора остановиться. Устроим этот сигнал, например, следующим образом. Будем считать, что существует сигнальная функция $\chi(u)$ со следующими свойствами:

если $\chi(e(x, m_1, \ldots, \delta(m_{k-1}))) > 0$, то процесс следует продолжать

если $\chi(e(x, m_1, \ldots, \delta(m_{k-1}))) = 0$, то процесс останавливается и $m_k = \psi(e(x, m_1, \ldots, \delta(m_{k-1})))$ и есть искомое значение функции $\gamma(x)$: $\gamma(x) = m_k$

Итак,

(T) Функция $\gamma(x)$ сводится по Тьюрингу, или Т-сводится, к функции $\delta(x)$, если существуют частично-рекурсивные функции $\psi(u)$ (сводящая) и $\chi(u)$ (сигнальная) со следующими свойствами. Функция $\psi(u)$ задает последовательность

$$m_1 = \psi(2^x)$$
$$m_2 = \psi(2^x 3^{m_1} 5^{\delta(m_1)})$$
$$m_3 = \psi(2^x 3^{m_1} 5^{\delta(m_1)} 7^{m_2} 11^{\delta(m_2)})$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$m_k = \psi(e(x, m_1, \delta(m_1), \ldots, m_{k-1}\ \delta(m_{k-1})))$$

Имеет место равенство

$$\gamma(x) = m_r$$

где r - наименьшее число такое, что

$$\chi(e(m_1, \delta(m_1), \ldots, m_{r-1}, \delta(m_{r-1}))) = 0$$

Кроме того, рассмотрим еще следующие три опреде-

ления сводимости.

(R) Функция $\gamma(x)$ рекурсивно сводится, или R-сводится к функции $\delta(x)$, если $\gamma(x)$ содержится в рекурсивном замыкании функции $\delta(x)$.

(A) Функция $\gamma(x)$ алгоритмически сводится, или A- сводится, к функции $\delta(x)$, если функция от комплексов $\Gamma(\bar{X})$ алгоритмически сводится к функции от комплексов $\bar{\Delta}(\bar{X})$ x/

(C) Функция $\gamma(x)$ канонически сводится, или C-сводится, к функции $\delta(x)$, если существует примитивно-рекурсивная функция $h(u, v, w)$ такая, что выполняются равенства

1) $g(x, 0) = x$
$g(x, m+1) = h(g(x, m), \varphi(m), \delta(\varphi(m)))$

2) $\gamma(x) = \tau(g(x, \mu m[\omega(g(x, m)) = 0]))$

где τ и ω - некоторые вполне определенные примитивно-рекурсивные функции (см. Обозначения), а $\varphi(m)$ - примитивно-рекурсивная функция, пересчитывающая, может быть с повторениями, область определения $\delta(x)$.

Все эти определения касаются функций от одного аргумента. Чтобы перенести их на случай большего числа аргументов поступим так. Отнесем каждой функции $f(x_1, \dots, x_n)$ ее одноместный представитель $f^*(u)$ так, чтобы

---

x/ Алгоритмическая сводимость функций от комплексов определялась во Введении. О взаимоотношении между $\gamma(x)$ и $\Gamma(\bar{X})$ и между $\delta(x)$ и $\bar{\Delta}(\bar{X})$ - см.Обозначения.

$$f^*(e(x_1,\dots,x_n)) = f(x_1,\dots,x_n)$$

где $e(x_1,\dots x_n) = p_0^{x_1+1} p_1^{x_2+1} \dots p_{n-1}^{x_n+1}$

Будем считать, что каждая функция по определению сводится во всех перечисленных смыслах к своему одноместному представителю и обратно. Поэтому нам достаточно иметь дело только с одноместными функциями.

Мы покажем эквивалентность всех этих определений сводимости. Сделаем это по схеме:

$$C \to R \to T \to A \to C$$

2.

Если $\gamma(x)$ C- сводится к $\delta(x)$, то и подавно $\gamma(x)$ R - сводится к $\delta(x)$.

3.

Пусть функция $\gamma(x)$ R- сводится к $\delta(x)$. Надо показать, что имеет место Т-сводимость.

Очевидно, что сама функция $\delta(x)$ и все примитивно-рекурсивные функции Т-сводятся к $\delta(x)$. Далее введем индуктивное построение. Пусть некоторые функции $\varphi_1, \varphi_2, \dots, \varphi_k$ Т-сводятся к $\delta(x)$. Тогда для них существуют сводящие функции $\psi_{\varphi_1}, \dots, \psi_{\varphi_k}$ и сигнальные функции $\chi_{\varphi_1}, \dots, \chi_{\varphi_k}$. Пусть теперь некоторая

функция f получается из $\varphi_1, \dots, \varphi_k$ рекурсивной операцией. Нужно построить для f сводящую функцию ψ_f и сигнальную χ_f.

Мы не будем проводить этого построения для всех рекурсивных операций, а укажем, как построить сводящую и сигнальную функции на примере одной рекурсивной операции - применения оператора μ.

Пусть функция $\theta(x,y)$ T-сводится к $\delta(x)$. Построим $\theta^*(u)$ такую, что $\theta^*(c(x,y)) = \theta(x,y)$ (т.-е., попросту), $\theta^*(2^x 3^y) = \theta(x,y)$. По определению существуют сводящая функция $\psi(u)$ и сигнальная $\chi(u)$ для $\theta^*(u)$. Надо построить функции $\tilde{\psi}(u)$ и $\tilde{\chi}(u)$ для функции

$$\mu(x) = \mu y\,[\theta(x,y) = 0]$$

Вычисление $\mu(x)$ идет так. Выписывается последовательность

$$\theta(x,0); \quad \theta(x,1); \quad \theta(x,2); \dots$$

до тех пор, пока первый раз не будет $\theta(x,y) = 0$. Тогда $\mu(x) = y$.

Вычислим $\theta(x,0) = \theta^*(2^x)$. Имеем схему (ведь θ^* T-сводится к $\delta(x)$):

$$\begin{matrix} 2^x & m_1^{(0)} & \delta(m_1^{(0)}) & m_2^{(0)} & \delta(m_2^{(0)}) & m_3^{(0)} & \dots & \delta(m_{r-1}^{(0)}) & m_r^{(0)} \\ & \chi_1^{(0)} & & \chi_2^{(0)} & & \chi_3^{(0)} & & & \chi_r^{(0)} \end{matrix}$$

где $\chi_k^{(0)} = \chi(c(2^x, m_1, \dots, m_{k-1}, \delta(m_{k-1})))$

Вычисления идут до тех пор, пока не будет в первый раз $\chi_r^{(0)} = 0$. Тогда $\theta(x,0) = \theta^*(2^x) = m_r^{(0)}$. Если

при этом $m_r^{(0)} = 0$, то весь процесс прекращается и $\mu(x) = 0$. Если $m_r^{(0)} \neq 0$, то переходим к вычислению $\Theta(x,1) = \Theta^*(2^x \cdot 3)$ по аналогичной схеме

$$\begin{matrix} 2^x \cdot 3 & m_1^{(1)} & \delta(m_1^{(1)}) & m_2^{(1)} & \delta(m_2^{(1)}) & m_3^{(1)} & \ldots & \delta(m_{r_1-1}^{(1)}) & m_{r_1}^{(1)} \\ & & \chi_1^{(1)} & & \chi_2^{(1)} & & \chi_3^{(1)} & & \chi_{r_1}^{1} \end{matrix}$$

до тех пор, пока не будет $\chi_{r_1}^{(1)} = 0$. Тогда

$$\Theta(x,1) = \Theta^*(2^x \cdot 3) = m_{r_1}^{(1)}$$

И так далее.

Таким образом, вычисление $\mu(x)$ идет по схеме

$$(\Sigma) \quad \begin{matrix} x & 2^x & m_1^{(0)} & \delta(m_1^{(0)}) & m_2^{(0)} & \ldots & m_{r_0}^{(0)} & 2^x \cdot 3 & m_1^{(1)} & \delta(m_1^{(1)}) & m_2^{(1)} \ldots m_{r_1}^{(1)} \\ & & & \chi_1^{(0)} & & \chi_2^{(0)} \ldots & \chi_{r_0}^{(0)} = 0 & & & \chi_1^{(1)} & \chi_2^{(1)} \ldots \chi_{r_1}^{(1)} = 0 \end{matrix}$$

$$\begin{matrix} \ldots\ldots & 2^x \cdot 3^y & m_1^{(y)} & \delta(m_1^{(y)}) & m_2^{(y)} \ldots m_{r_y}^{(y)} & \ldots\ldots \\ & & & \chi_1^{(y)} & \chi_2^{(y)} \quad \chi_{r_y}^{(y)} = 0 & \ldots \end{matrix}$$

Итак, мы указали алгоритм, строящий последовательность (Σ). Построение идет до тех пор, пока не будет одновременно

$$(*) \quad \begin{cases} \chi_{r_y}^{(y)} = 0 \\ m_{r_y}^{(y)} = 0 \end{cases}$$

тогда $\mu(x) = y$

Каждый член последовательности (Σ), не входящий под знак функции δ, эффективно строится по предыдущей строке.

Ясно, что алгоритм построения последовательности (Σ) можно записать в виде некоторой частично-рекурсивной функции. Ясно также, что можно построить частично-рекурсив-

ную функцию, прекращающую построение этой последовательности, как только выполняются равенства (*).

Это и будут искомые сводящая и сигнальная функции для Т-сводимости $\mu(x)$ к $\delta(x)$.

4.

Пусть функция $\gamma(x)$ Т-сводится к функции $\delta(x)$. Покажем, что тогда имеет место и алгоритмическая сводимость $\gamma(x)$ к $\delta(x)$ т.-е. алгоритмическая сводимость $\bar{\Gamma}(\bar{X})$ к $\bar{\Delta}(\bar{X})$

В самом деле, Т-сводимость $\gamma(x)$ к $\delta(x)$ задается сводящей функцией $\psi(n)$ и сигнальной $\chi(n)$. Построим машины Колмогорова $\mathfrak{T}_\psi$ и $\mathfrak{T}_\chi$ вычисляющие функции $\bar{\Psi}(\bar{N})$ и $\bar{\mathcal{X}}(\bar{N})$, которые изображают $\psi(n)$ и $\chi(n)$ Построим также машину $\mathfrak{T}_e$, которая комплекс

$$\bar{X}_1 - \bar{X}_2 - \ldots - \bar{X}_n$$

перерабатывает в комплекс, являющийся изображением числа $e(x_1, \ldots, x_n)$. Если область определения $\delta(x)$ пересчитывается рекурсивной функцией $\varphi(m)$ (а мы только такие функции $\delta(x)$ и рассматриваем), то область определения функции $\bar{\Delta}(\bar{X})$ пересчитывается алгоритмической функцией $\bar{\Phi}(\bar{M})$. Построим бесконечный комплекс $\mathfrak{K}_{\bar{\Phi}}(\bar{\Delta})$ (см. стр. 26). Теперь надо алгоритмически свести $\bar{\Gamma}(\bar{X})$ к $\bar{\Delta}(\bar{X})$, т.-е. построить машину, осуществляющую сводящий алгоритм $\Pi_{\bar{\Phi}}$.

Иными словами, надо построить машину $\mathcal{T}$ с начальным состоянием $\mathcal{K}_{\bar{\Phi}}(\bar{\Delta})$ осуществляющую функцию. $\bar{\Gamma}(\bar{X})$.

Задавать эту машину явными формулами было бы слишком громоздко, но мы опишем ее действие, из чего будет ясно, как ее построить.

Итак, вот действие $\Pi_{\bar{\Phi}}(\bar{X})$. Комплекс $\bar{X}$ прежде всего подвергается действию машины $\mathcal{T}_e$, которая вырабатывает $\bar{N}_1$ ($\bar{N}_1$ - изображение числа $n_1 = e(x)$) Затем к $\bar{N}_1$ применяется $\mathcal{T}_\psi$ и получается $\bar{M}_1$ (изображение m_1). Затем в верхнем ряду комплекса $\mathcal{K}_{\bar{\Phi}}(\bar{\Delta})$ x/ ищется первый комплекс P_k^+ такой, что соответствующий П-комплекс P_k совпадает с $\bar{M}_1$ (машину для просмотра верхнего ряда $\mathcal{K}_{\bar{\Phi}}(\bar{\Delta})$ построить нетрудно). Когда такой P_k^+ обнаружится, берется соответствующий ему комплекс Q_k^+ в нижнем ряду и далее П-комплекс Q_k. Этот П-комплекс Q_k и есть $\bar{\Delta}(\bar{M}_1)$ (изображение числа $\delta(m_1)$). После этого образуется комплекс $\mathcal{U}_1$

$$\bar{X} - \bar{M}_1 - \bar{\Delta}(\bar{M}_1)$$

К нему применяется машина $\mathcal{T}_e$, перерабатывающая его в некоторый комплекс $\bar{N}_2$. ($\bar{N}_2$ - изображение числа $n_2 = e(x, m_1, \delta(m_1))$). Затем к $\bar{N}_2$ применяется $\mathcal{T}_\psi$ и получается $\bar{M}_2$. Затем снова просматривается верхний ряд $\mathcal{K}_{\bar{\Phi}}(\bar{\Delta})$ пока не дойдем до первого P_k^+, для которого P_k совпадает с $\bar{M}_2$; для него находим в нижнем ряду

----

x/ "Верхний ряд" комплекса $\mathcal{K}_{\bar{\Phi}}(\bar{\Delta})$ образуют комплексы $P_1^+, P_2^+, \ldots, P_m^+, \ldots$ (см.стр. 26).

соответствующий $Q_k = \bar{\Delta}(P_k) = \bar{\Delta}(\bar{M}_2)$, далее образуем комплекс U_2, присоединяя к U_1 комплексы $\bar{M}_2$ и $\bar{\Delta}(\bar{M}_2)$. К U_2 применяем $\mathcal{T}_e$, получаем $\bar{N}_2$. Наконец, к $\bar{N}_2$ применяем $\mathcal{T}_\psi$ и получаем $\bar{M}_3$. И так далее.

Одновременно к каждому $\bar{N}_k$ применяется машина $\mathcal{T}_\chi$. Процесс идет до тех пор, пока не будет $\bar{\mathcal{X}}_e(\bar{N}_k) = \bar{0}$ ($\bar{0}$ - изображение числа 0). Тогда процесс прекращается, и соответствующей комплекс $\bar{M}_k$ и есть ответ:

$$\Pi_{\bar{\Phi}}(\bar{X}) = \bar{M}_k$$

Ясно, что можно построить машину, которая бы производила указанные операции.

5.

Осталось показать, что если $\gamma(x)$ А-сводится к $\delta(x)$, то она и **С** — сводится к $\delta(x)$).

Доказательство основывается на следующей лемме

Лемма. Пусть некоторая функция от комплексов $\bar{\Gamma}(K)$ алгоритмически сводится к некоторой другой функции от комплексов $\bar{\Delta}(K)$ при помощи сводящего условного алгоритма Π_Θ.
Тогда существует такая примитивно-рекурсивная функция $\sigma(u, v, w)$, что индуцированная в натуральном ряду функция $\bar{\gamma}(k)$ вычисляется по формуле

$$\rho(k,0) = \xi(k)$$
$$(4.01) \quad \rho(k,m+1) = \sigma(\rho(k,m), \theta(v(m)), \bar{\delta}(\theta(v(m))))$$
$$\bar{\gamma}(k) = \xi(\rho(k, \mu m[\omega(\rho(k,m)) = 0]))$$

(4,01)

Здесь $\zeta, \xi, \check{\nu}$ и ω - фиксированные примитивно-рекурсивные функции (см. Обозначения), $\bar{\delta}(k)$ - функция, индуцированная в натуральном ряду функцией $\bar{\Delta}(k)$, а $\theta(m)$ -функция, индуцированная в натуральном ряду функцией от комплексов $\Theta(\bar{M})$, пересчитывающей область определения $\bar{\Delta}(k)$.

Предположим, что эта лемма уже доказана и пусть $\gamma(x)$ А-сводится к $\delta(x)$, т.-е. $\bar{\Gamma}(\bar{X})$ алгоритмически сводится к $\bar{\Delta}(\bar{X})$. Покажем, что $\gamma(x)$ С-сводится к $\delta(x)$.

Область определения $\delta(x)$ пересчитывается примитивно-рекурсивной функцией $\varphi(m)$. Эта функция изображается функцией от комплексов $\bar{\Phi}(\bar{M})$, пересчитывающей область определения $\bar{\Delta}(\bar{X})$. Как отмечалось во Введении факт сводимости $\bar{\Gamma}(\bar{X})$ к $\bar{\Delta}(\bar{X})$ не зависит от выбора функции Θ, для которой строится сводящий алгоритм Π_{Θ}. (Это показано в начале § 5.) Поэтому мы можем считать, что $\bar{\Gamma}(\bar{X})$ сводится к $\bar{\Delta}(\bar{X})$ именно посредством алгоритма $\bar{\Phi}$ с начальным состоянием $\mathfrak{K}_{\bar{\Phi}}(\bar{\Delta})$.

Применим теперь к алгоритмической сводимости $\bar{\Gamma}(\bar{X})$ к $\bar{\Delta}(\bar{X})$ нашу лемму. В формулах (4,01) надо число k заменить на число $\bar{x}$ - номер комплекса $\bar{X}$, изображающего число x. Этот номер есть примитивно-рекурсивная функция от x:

$$\bar{x} = \nu(x)$$

Далее, так как роль Θ играет теперь $\bar{\Phi}$, то вместо θ

надо подставить $\bar{\varphi}$.

Учитывая все это, получим из (4,01):

$$\rho(v(x),0) = \zeta(v(x))$$

(4,02) $$\rho(v(x),m+1) = \sigma(\rho(v(x),m), \bar{\varphi}(v(m)), \bar{\delta}(\bar{\varphi}(v(m))))$$

$$\bar{\gamma}(v(x)) = \xi(\rho(v(x), \mu m[\omega(\rho(v(x),m))=0]))$$

Заметим, что

$$\bar{\gamma}(v(x)) = v(\gamma(x))$$

В самом деле, по построению

$$y = \gamma(x)$$

$$\bar{Y} = \Gamma(\bar{X})$$

$$(\text{номер } \bar{Y}) = \bar{\gamma}(\text{номер } \bar{X})$$

$$v(y) = \bar{\gamma}(v(x))$$

$$v(\gamma(x)) = \bar{\gamma}(v(x))$$

Точно также

$$\bar{\varphi}(v(m)) = v(\varphi(m)) \; ; \qquad \bar{\delta}(v(u)) = v(\delta(u))$$

Кроме того, учтем, что ζ и v взаимно обратны (см. Обозначения). Поэтому из (4,02) получим

$$\rho(v(x),0) = x$$

(4,03) $$\rho(v(x),m+1) = \sigma(\rho(v(x),m), v(\varphi(m)), v(\delta(\varphi(m))))$$

$$v(\gamma(x)) = \xi(\rho(v(x), \mu m[\omega(\rho(v(x),m))=0]))$$

К последнему из равенств (4,03) применим примитивно-рекурсивную функцию ζ , обратную к v .

Полагая

$$\rho(v(x),m) = g(x,m)$$

$$\sigma(u, v(v), v(w)) = h(u,v,w)$$

$$\zeta(\xi(u)) = \tau(u)$$

получим окончательно.

$$(4,04)\quad \begin{aligned} & g(x,0)=x \\ & g(x,m+1)=h(g(x,m),\varphi(m),\delta(\varphi(m))) \\ & \gamma(x)=\tau(g(x,\mu m[\omega(g(x,m))=0])) \end{aligned}$$

Равенства (4,04) и означают С-сводимость $\gamma(x)$ к $\delta(x)$.

Итак, осталось доказать лемму. Пусть условный алгоритм Π_Θ с начальным состоянием $\mathfrak{K}_\Theta(\bar\Delta)$ осуществляет функцию $\Gamma(K)$. Надо показать, что выполняются равенства (4,01).

Мы произведем то же изменение в определении машины Колмогорова, что и в § 3, а именно, будем считать, что конец процесса наступает, при появлении знака ω на активной вершине; связная компонента получившегося комплекса - есть решение.

П-комплекс $\mathfrak{K}_\Theta(\bar\Delta)$ обозначим для простоты $\mathfrak{K}$. Его можно представить в виде об'единения двух комплексов $\mathfrak{L}_m$ и $\mathcal{L}_m$. При этом $\mathfrak{L}_m$ - конечный комплекс вида

а $\mathcal{L}_m$ - остаточный бесконечный комплекс вида

43

Подлежащий переработке комплекс K присоединяется к $\mathfrak{K}$ посредством вершины e_{-1}. Получается комплекс $\mathfrak{K}^1$. Он перерабатывается в $\mathfrak{K}^2$, далее в $\mathfrak{K}^3$ и т.д.

Вначале, в комплексе $\mathfrak{K}^1$, потенциальная часть охватывает, помимо вершин из K, лишь вершины e_{-1} и e_0, т.-е. вся она умещается в пределах $\mathfrak{L}_1$. Далее за каждый шаг потенциальная часть передвигается не более, чем на один отрезок, поэтому в комплексе $\mathfrak{K}^2$ переработка не выведет нас за пределы $\mathfrak{L}_2$. Вообще, через m шагов переработка затронет лишь комплекс $\mathfrak{L}_m$.

Таким образом, если комплекс $\mathfrak{K}^1$ есть об'единение комплексов $\mathfrak{L}_m^1$ и $\mathfrak{L}_m^1$ x/, то $\mathfrak{K}^m$ есть об'единение комплексов $\mathfrak{L}_m^m$ и $\mathfrak{L}_m^m$, где $\mathfrak{L}_m^1$ и $\mathfrak{L}_m^m$ совпадают.

Более наглядно. Перед началом переработки был комплекс $\mathfrak{K}^1$ вида

$$\underbrace{K - \mathfrak{L}_m}_{\mathfrak{L}_m^1} - \underbrace{\mathfrak{L}_m}_{\mathfrak{L}_m^1}$$

Через m шагов он превратился в комплекс $\mathfrak{K}^m$ вида

$$\mathfrak{L}_m^m - \mathfrak{L}_m^m$$

причем переработка коснулась только комплекса $\mathfrak{L}_m^1$, который переработался в $\mathfrak{L}_m^m$, а $\mathfrak{L}_m^1$ остался неизменным: $\mathfrak{L}_m^1 = \mathfrak{L}_m^m$.

Таким образом, сами комплексы $\mathfrak{K}^1, \mathfrak{K}^2, \dots, \mathfrak{K}^m, \dots$ играют роль вспомогательного материала, из которого конструируется последовательность $\mathfrak{L}_1, \mathfrak{L}_2^2, \dots, \mathfrak{L}_m^m$

---

x/ $\mathfrak{L}_m^1$ есть результат присоединения K и $\mathfrak{L}_m$.
$\mathfrak{L}_m^1$ - совпадает с $\mathfrak{L}_m$

Эта последовательность строится до тех пор, до того первого m, пока на активной вершине $\mathcal{L}_m^m$ не возникнет сигнал ω. Связная компонента этой вершины (которая, легко видеть, должна лежать внутри $\mathcal{L}_m^m$, а иначе она бесконечна) и есть решение.

Вычислим номер решения. Обозначим номер комплекса $\mathcal{L}_m^m$ ЧЕРЕЗ $\rho(k,m)$ ~~через~~ (k - номер K). Посмотрим, как ~~получается из~~ $\mathcal{L}_{m+1}^{m+1}$ получается из $\mathcal{L}_m^m$ $(m \geq 1)$

Комплекс $\mathcal{K}^m$ имеет вид

$\mathcal{L}_m^m$ P_m^+ P_{m+1}^+ Q_m^+ Q_{m+1}^+ ...

Комплекс $\mathcal{K}^{m+1}$ имеет вид

$\mathcal{L}_{m+1}^{m+1}$ P_{m+1}^+ P_{m+2}^+ Q_{m+1}^+ Q_{m+2}^+

95.

Таким образом, в образовании комплекса $\mathcal{L}^{m+1}_{m+1}$ участвовали лишь комплекс $\mathcal{L}^{m}_{m}$ и комплекс

$$\begin{matrix} (P^+_m) \\ | \\ o \\ | \\ (Q^+_m) \end{matrix}$$

Но этот последний комплекс сам получается из P_m и Q_m. Поэтому, в образовании $\mathcal{L}^{m+1}_{m+1}$ в конечном счете участвовали $\mathcal{L}^{m}_{m}$, P_m и Q_m. Так же, как и в § 3, очевидно существование примитивно-рекурсивной функции $\sigma(u,v,w)$, такой что

$$\rho(k,m+1) = \sigma(\rho(k,m), p_m, q_m) \qquad (m \geq 1)$$

Здесь p_m и q_m соответственно номера комплексов P_m и Q_m. Положим $p_0 = q_0 = 0$.

При этом $\rho(k,1) = b(k)$, где $b(k)$ - номер комплекса $\mathcal{L}^{1}_{1}$:

$$K \text{---} \underset{e_{-1}}{o} \text{---} \underset{e_0}{o}$$

Всегда можно считать, что $b(k) = \sigma(\varsigma(k), p_0, q_0)$ (ς - фиксированная примитивно-рекурсивная функция, восстанавливающая число по номеру его изображения, см. Обозначения).

Кроме того, так как $Q_m = \Delta(P_m)$, то $q_m = \delta(p_m)$

Далее $P_m = \Theta(\overline{M})$, откуда

$$p_m = \theta(\bar{m}) = \theta(\nu(m))$$

Окончательно, $\rho(k,m)$ задается так:

$$\rho(k,0) = \varsigma(k)$$

$$\rho(k,m+1) = \sigma(\rho(k,m), \theta(\nu(m)), \delta(\theta(\nu(m))))$$

Процесс продолжается до тех пор, пока не возникнет знак ω, т.-е. пока номер $\rho(k,m)$ не будет удовлетво-

рять равенству

$$\omega(\rho(k,m))=0$$

Итак, процесс идет до числа m_0, определяемого условием

$$m_0=\mu m[\omega(\rho(k,m))=0]$$

Тогда берется комплекс $\mathcal{L}^{m_0}_{m_0}$ с номером $\rho(k,m_0)$ и связная компонента этого комплекса с номером $\xi(\rho(k,m_0))$

Эта связная компонента и есть решение, а ее номер есть номер решения.

Таким образом, если функция $\bar{\Gamma}(k)$ сводится к $\bar{\Delta}(k)$ при помощи сводящего алгоритма Π_{Θ}, то существует примитивно-рекурсивная функция $\sigma(u,v,w)$, что выполняются равенства

$$\rho(k,0)=\jmath(k)$$
$$\rho(k,m+1)=\sigma(\rho(k,m),\theta(\nu(m)),\bar{\delta}(\theta(\nu(m))))$$
$$\bar{\gamma}(k)=\xi(\rho(k,\mu m[\omega(\rho(k,m))=0]))$$

Утверждение леммы доказано.

6.

Итак, если функция $\gamma(x)$ сводится к функции $\delta(x)$ (доказав эквивалентность всех определений сводимости, мы можем говорить просто "сводится") и если область определения $\delta(x)$ совпадает с множеством значений $\varphi(m)$, то существует примитивно-рекурсивная функция $\sigma(u,v,w)$, такая что

(4,05)
$$g(x,0)=x$$
$$g(x,m+1)=h(g(x,m),\varphi(m),\delta(\varphi(m)))$$
$$\gamma(x)=\tau(g(x,\mu m[\omega(g(x,m))=0]))$$

В частном случае, когда $\delta(x)$ всюду определена, можно положить $\varphi(m)=m$ и равенства (4,05) перепишутся в виде

$$g(x,0)=x$$
$$g(x,m+1)=h(g(x,m),\ m,\ \delta(m))$$
$$\gamma(x)=\tau(g(x,\mu m[\omega(g(x,m))=0]))$$

В другом частном случае, когда $\delta(x)$ - примитивно-рекурсивная функция, получается, что $g(x,m)$ тоже примитивно-рекурсивна и мы приходим к равенству

$$\gamma(x)=\tau(g(x,\mu m[\omega(g(x,m))=0]))$$

в полном согласии с последней формулой § 3.

§ 5. Дополнительные сведения об алгоритме Колмогорова.

1.

В этом пункте мы покажем, что факт сводимости или не сводимости функции от комплексов $\Gamma(K)$ к функции от комплексов $\Delta(K)$ не зависит от выбора функции φ, разворачивающей в перечислимую последовательность область определения $\Delta(K)$.

Пусть $\varphi(\overline{M})$ и $\Theta(\overline{M})$ - две функции, каждая из которых разворачивает область определения $\Delta(K)$ в перечислимую последовательность. Пусть условный алгоритм Π_φ сводит $\Gamma(K)$ к $\Delta(K)$. Построим условный алгоритм Π_Θ, сводящий $\Gamma(K)$ к $\Delta(K)$.

Чтобы построить Π_Θ, надо сперва построить его начальное состояние $\mathfrak{K}_\Theta(\Delta)$. Опишем теперь, как действует Π_Θ.

Имея в своем распоряжении во-первых $\mathfrak{K}_\Theta(\Delta)$ и во-вторых алгоритм для построения $\varphi(\overline{M})$ (ибо φ -функция алгоритмическая), мы можем задать правила, осуществляющие построение $\mathfrak{K}_\varphi(\Delta)$.

Кроме того, ведь у нас есть правила, вычисляющие $\Gamma(K)$ по $\mathfrak{K}_\varphi(\Delta)$. Поэтому нетрудно написать правила, которые осуществляли бы вычисление $\Gamma(K)$ основываясь

43

не на готовом, а на непрерывно формирующемся комплексе $K_{\Phi}(\Delta)$.

2.

Приведем без доказательства следующие теоремы:

Т е о р е м а   п е р е с ч е т а. Для каждого порядка (n,α) существует алгоритм $I_{(n,\alpha)}$, применимый ко всякому комплексу K порядка (n,α) и перерабатывающий этот комплекс в комплекс $\overline{K}$ - изображение его номера k.

Т е о р е м а   в о с с т а н о в л е н и я. Для каждого порядка (n,α) существует алгоритм $\Lambda_{(n,\alpha)}$, применимый ко всякому комплексу $\overline{K}$, являющемуся изображением номера k некоторого комплекса K порядка (n,α), и перерабатывающий $\overline{K}$ в K

Эти теоремы позволяют строить многие конкретные алгоритмы. Как, например, построить алгоритм, распознающий равенство произвольных П-комплексов K_1 и K_2 порядка (n,α)? Применим к ним обоим алгоритм $I_{(n,\alpha)}$. Мы получим комплексы $\overline{K}_1$ и $\overline{K}_2$, которые будут равны или неравны одновременно с K_1 и K_2. В то же время распознать равенство $\overline{K}_1$ и $\overline{K}_2$ уже просто.

Применим эти теоремы к доказательству того, что область определения всякой алгоритмической функции

перечислима. Пусть дана алгоритмическая функция $\Gamma(K)$. Мы рассматриваем эту функцию лишь на множестве комплексов некоторого фиксированного порядка (n,α). Функция от комплексов (n,α) индуцирует в натуральном ряду функцию $\gamma(k)$. Так как $\gamma(k)$ частично-рекурсивна, то область ее определения рекурсивно-перечислима (это известный факт), она перечисляется рекурсивной функцией $k=\varphi(m)$. Эта функция изображается функцией от комплексов

$$\bar{K} = \bar{\Phi}(\bar{M})$$

Применим к обоим частям этого равенства алгоритм $\Lambda_{(n,\alpha)}$. Получим

$$K = \Lambda_{(n,\alpha)} \Phi(\bar{M})$$

Алгоритм $\Lambda_{(n,\alpha)}\Phi$ и есть алгоритм, перечисляющий область определения $\Gamma(K)$.

3.

Безусловный алгоритм Γ задается правилами переработки. Эти правила нетрудно записать в виде некоторого комплекса S_Γ с номером s_Γ. Комплекс S_Γ будем называть записью алгоритма Γ.

Легко обнаружить существование универсальной частично-рекурсивной функции $\alpha(s_\Gamma, k)$, такой что если s_Γ - номер записи алгоритма Γ, а k - номер

комплекса K , то $\ell=\alpha(s_r,K)$ есть номер комплекса $L=\Gamma(K)$.

Отсюда следует существование универсального алгоритма $\Upsilon_{(n,\alpha)}$, который, будучи применен к комплексу, составленному из S_r и K дает $L=\Gamma(K)$ (для K и S_r порядка (n,α)). В самом деле, применив к S_r и K алгоритм $I_{(n,\alpha)}$, получим $\bar{S}_r$ и $\bar{K}$. Из существования функции $\ell=\alpha(s_r,K)$ легко вывести существование такой алгоритмической функции $\bar{A}$, которая, будучи применена к об'единению $\bar{S}_r$ и $\bar{K}$, давала бы $\bar{L}$. После этого, применив к $\bar{L}$ алгоритм $\Lambda_{(n,\alpha)}$, получим L .

4.

Каждый алгоритм мы рассматриваем лишь в применении к комплексам некоторого фиксированного порядка ((n,α)). Комплексы порядка (n,α) образуют систему $\sum(n,\alpha)$. Про алгоритм, определенный на некотором множестве комплексов системы $\sum(n,\alpha)$ мы скажем, что он определен <u>в</u> системе $\sum(n,\alpha)$. Очевидно, $\sum(n_1,\alpha_1)\subseteq\sum(n_2,\alpha_2)$ если порядок (n_1,α_1) ниже, чем порядок (n_2,α_2) , т.-е. если

$n_1\leq n_2,\quad \alpha_1\leq\alpha_2$

Алгоритм, определенный в некоторой системе, содержащей $\sum(n,\alpha)$ назовем алгоритмом <u>над</u> $\sum(n,\alpha)$. Можно построить теорию перевода комплексов и алгоритмов из системы более высокого порядка в системы более низкого порядка,

аналогично теории А.А.Маркова [1] о переводе слов и алгоритмов из одного алфавита в другой. При этом роль двухбуквенного алфавита $\{a, b\}$ будет играть система $\sum(2,3)$.

Теория перевода и существование универсального алгоритма позволяют обнаружить несуществование некоторых алгоритмов Колмогорова, дословно так же, как обнаруживается невозможность нормальных алгорифмов Маркова, приведенных в [1] .

Однако, следует отметить, что несуществование этих алгоритмов видно и непосредственно из теории рекурсивных функций.

5.

В заключение приведем некоторое видоизменение изученного определения алгоритма, также принадлежащее А.Н. Колмогорову. Новое определение во всем совпадает со старым, и отличается от старого лишь пониманием обозримой части. В старом определении обозримой частью являлся по существу потенциальный подкомплекс; в новом это не так.

Машина перерабатывает комплексы, с выделенной основной вершиной. Обозримой частью считается все то, что достижимо цепями длины ≤ 7 от основной вершины. За один шаг обозримая часть перерабатывается по

заданным правилам переработки.

В остальных деталях новое определение совпадает с первоначальным.

Легко проверить эквивалентность обоих определений.

ЛИТЕРАТУРА

[1] А. А. Марков, Теория алгорифмов, («Труды Математического Института им. В.А.Стеклова» XXXVIII, М., 1951, стр. 176 - 189).

[2] A. Church, An unsolvable problem of elementary number theory. («American Journal of Mathematics», vol. 58 (1936), pp. 345-363)

[3] S.C. Kleene, Recursive predicates and quantifiers. («Transactions of the American Mathematical Society», vol. 53 (1943), pp. 41-73)

[4] E.L. Post, Finite combinatory processes - formulation I. («The Journal of Symbolic Logic», vol. 1 (1936), pp. 103-105)

[5] E.L. Post, Formal reduction of the general combinatorial decision problem, («American Journal of Mathematics», vol. 65 (1943) pp. 197-215)

[6] E.L. Post, Recursively enumerable sets of positive integers and their decision problems. («Bulletin of the American Mathematical Society» vol. 5 (1944), pp. 284-316)

[7] R.M. Robinson, Primitive recursive functions («Bulletin of the American Mathematical Society» vol. 53 (1947), pp. 925-942

[8] B. Rosser, An informal exposition of proofs of Gödel theorems and Church's theorem. («The Journal of Symbolic Logic», vol. 4 (1939), pp. 53-60)

[9] A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem. («Proceeding of the London Mathematical Society», vol. 42 (1936-1937), pp. 230-265)

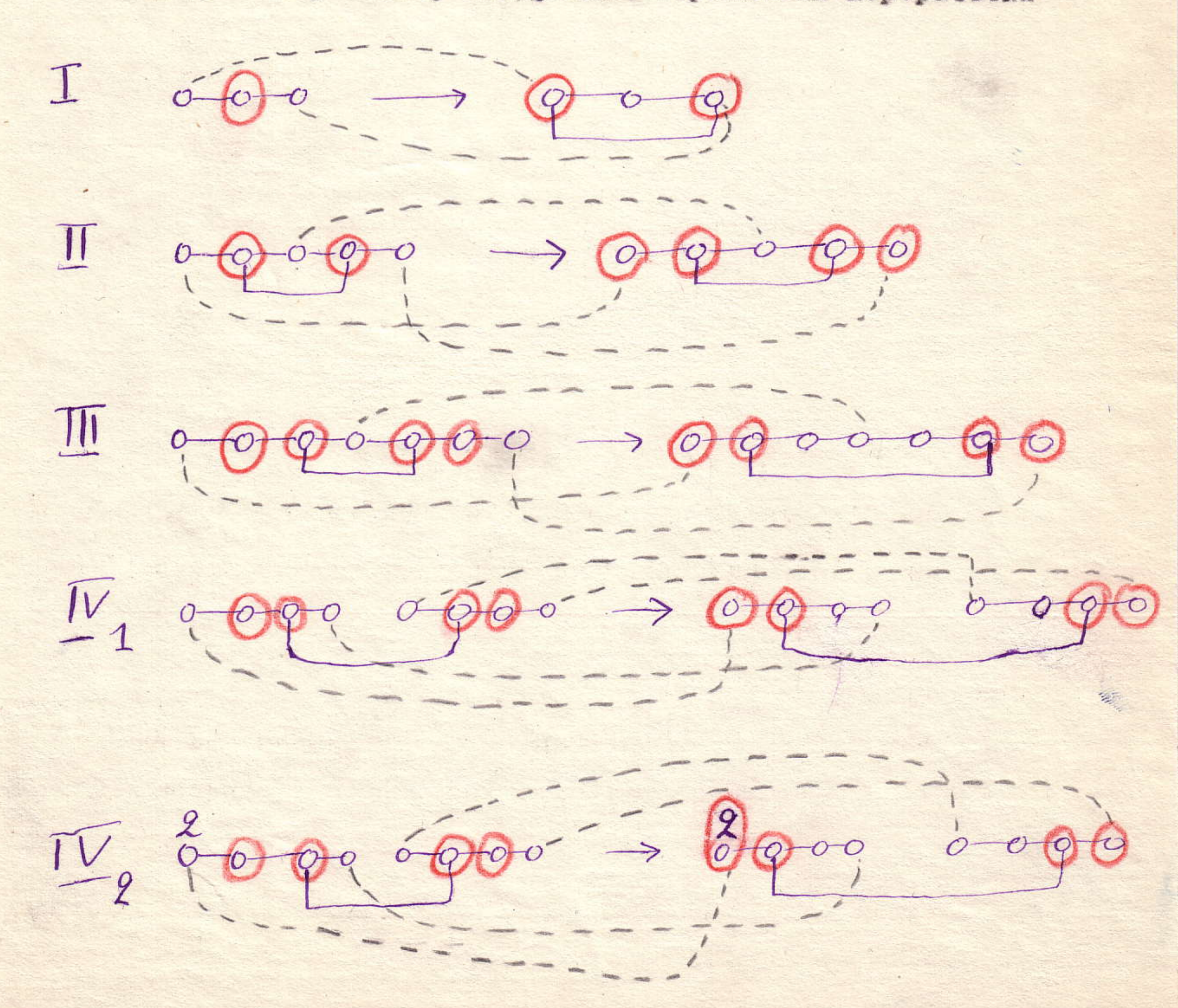
[10] A.M. Turing, Computability and λ-definability. («The Journal of Symbolic Logic», vol. 2 (1937), pp. 153-163)

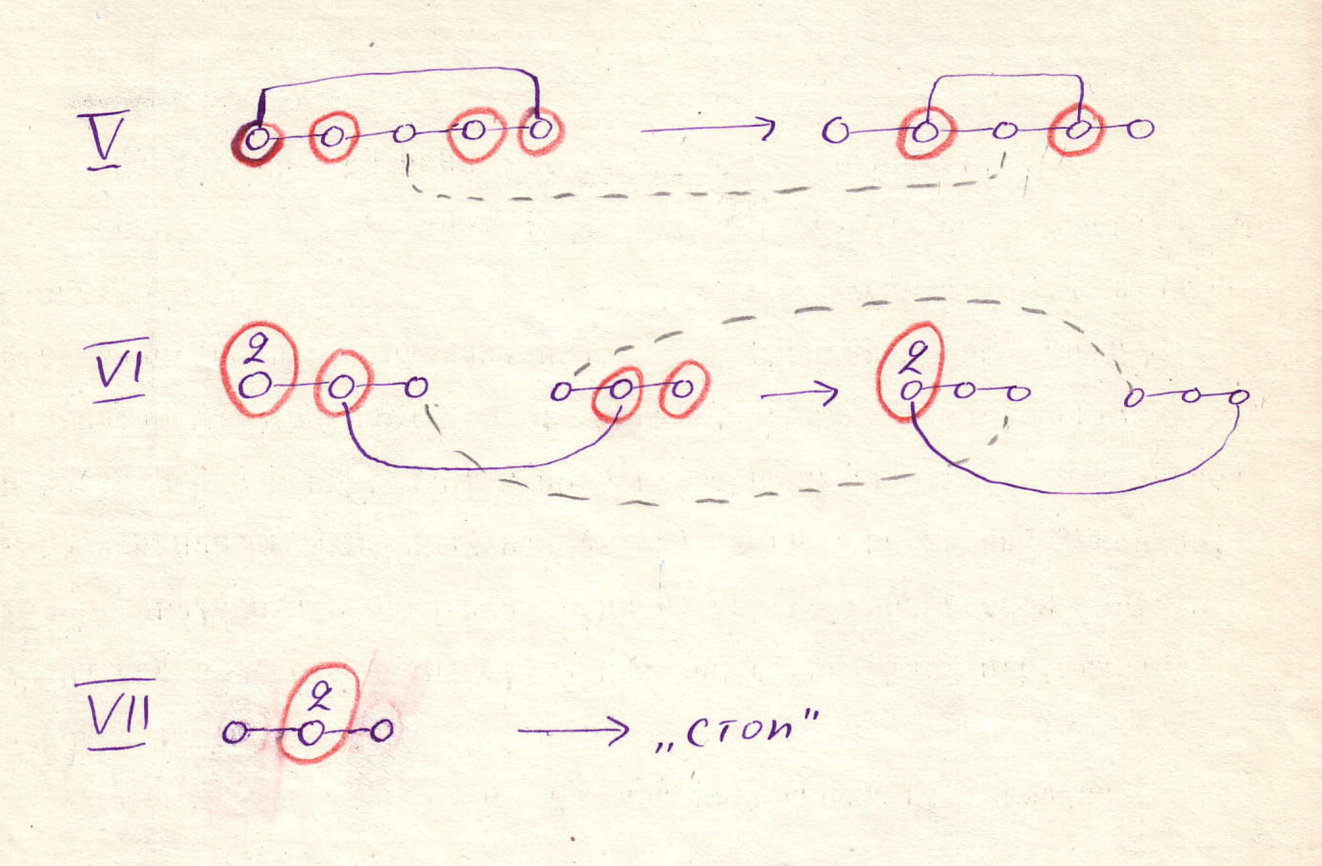
ПРИЛОЖЕНИЕ

Приведем пример простейшей машины Колмогорова.

Пары (H, H^*) из множества $\mathfrak{M}$ правил переработки будем писать в виде $H \to H^*$. Пунктирными линиями будем обозначать соответствие, установленное между граничными вершинами H и некоторыми вершинами H^*; пунктир, таким образом, показывает, в какие вершины переходят граничные вершины из H^*. При записи П-комплексов мы для упрощения опустим значки на концах отрезков; кроме того, если в некоторой вершине характеристическая функция принимает значение единица, то при записи мы будем эту единицу опускать.

Зададим нашу машину следующими правилами переработки





За начальное состояние машины примем пустое множество.

Применим построенный алгоритм к П-комплексу K :

Так как начальное состояние пусто, то комплекс K ни к чему не присоединяется и $K^0 = K$.

Потенциальный подкомплекс в K^0 имеет вид

По правилу I он заменяется на

Таким образом, возникает комплекс K^1 :

В K^1 потенциальный подкомплекс имеет вид

Прменяя правило II , получим K^2 :

Далее получим (применяя III)

K^3:

Применяя IV_2, получим

K^4:

И, наконец, применяя VI

K^5:

Потенциальный подкомплекс в K^5 имеет вид

Применяя правило VII , мы получим сигнал "стоп". П-комплекс K^5 есть решение.

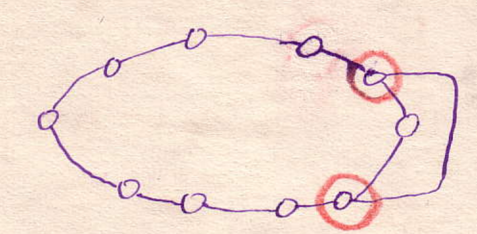
Подвергнем действию той же машины комплекс L :

Он преобразуется в L^1 :

Далее

L^2: L^3: L^4:

Потенциальный подкомплекс в L^4 имеет вид



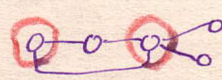
Применяя правило $\underline{V}$, получаем L^5:

Мы видим, что $L^5 = L^1$. Процесс, таким образом, циклически повторяется. Машина будет работать над L неограниченно. Остановка никогда не наступит.

Если, наконец, мы захотим применить алгоритм к комплексу



то машина переработает его в комплекс



и остановится безрезультатно.

Итак, одна и та же машина в применении к одним П-комплексам может давать результативный конец, в применении к другим - безрезультатную остановку, в применении к третьим - неограниченное продолжение процесса переработки.

П Р О Т О К О Л

заседания кафедры истории математических наук от 10 мая 1952г.

ПРИСУТСТВУЮТ: А.Н.Колмогоров, П.С.Новиков, С.А.Яновская, И.Г.Башмакова

СЛУШАЛИ: Защита студентом У-го курса В.А.Успенским дипломной работы на тему: Общее определение алгоритмической нчислимости и алгоритмической сводимости

Заслушав выступления студента В.А.Успенского, руководителя работы ак. А.Н.Колмогорова, рецензента П.С.Новикова и С.А.Яновской, кафедра ПОСТАНОВИЛА: признать работу В.А.Успенского выдающейся. Отметить, что работа содержит ряд значительных новых результатов и сидетельствует о глубоком владении автором всей трудной проблематикой теории алгоритмов. Отметить также прекрасное оформление работы и признать необходимым опубликование ее.

ЗАВ. КАФЕДРОЙ ПРОФ. С.Яновская (С.А.Яновская)

12 мая 1952

Московский ордена Ленина Государственный университет имени М. В. Ломоносова

_______________ факультет

кафедра _______________

ОТЗЫВ НА ДИПЛОМНУЮ РАБОТУ

студента ______ курса Успенского В. А.
(фамилия и инициалы)

Тема Общее определение алгоритмической вычислимости и алгоритмической сводимости.

Руководитель, ~~рецензент~~ академик Колмогоров А. Н.
(ученая степень, ученое звание, фамилия и инициалы)

В отзыве должны быть отмечены достоинства и недостатки дипломной работы.

Работа интересна в двух отношениях:

1/ В ней впервые подвергнута подробному исследованию алгоритмическая сводимость вычисления функции

$y = \gamma(x)$

к вычислению функции

$y = \delta(x)$

/ x и y натуральные числа/.

2/В ней подвергается более полному, чем до сих пор делалось, анализу само понятие алгоритмической вычислимости.

1/.Автор приводит только одно, предлагавшееся до него, формально безукоризненное определение алгорифмической сводимости, которое он на стр. 22 приписывает Б. А. Трахтенброту: функция γ "сводится" к функции δ, если γ принадлежит рекурсивному замыканию δ. Автор показывает, что в действительности такая сводимость может всегда быть осуществлена очень простым каноническим образом при помощи раз на всегда заданных примитивно-рекурсивных функций $\tau(u)$ и $\omega(u)$ и зависящих от пары γ, δ примитивно-рекурсивных функций $h(u,v,w)$ и $\varphi(m)$. См. об этом теорему на стр. 28. Это основной новый с чисто математической точки зрения результат работы.

Определение сводимости по Трахтенброту нуждается в известном "оправдании" его соответствия интуитивной идее сводимости в смысле

существования "механического" способа получения при любом x значения $\gamma(x)$ в предположении, что получение значений $\delta(x)$ сделано каким-то способом "доступным" для любого x. Общие контуры возможной формализации этой идеи были намечены Поустом. В дипломной работе полностью воспроизведен перевод соответствующего места статьи Поуста. Автор дипломной работы, повидимому впервые, дает соответствующее этой идее определение сводимости с полной отчетливостью и показывает его эквивалентность определению Трахтенброта. Это тоже весьма существенное достижение автора дипломной работы.

2/ Кроме того в работе содержится хороший обзор различных предлагавшихся ранее определений алгоритмической вычислимости числовой функции $y=\gamma(x)$. В центре изложения помещено определение, предложенное мною, интерес которого на мой взгляд убедительно аргументирован автором дипломной работы. Доказана равносильность этого определения прежде предлагавшимся. В известном смысле слова этот результат можно рассматривать как "обоснование" прежних определений, так как в моем определении становится особенно ясной основная идея алгоритмической вычислимости, которая отличается от вычислимости обыкновенным реальным счетным механизмом только неограниченным об"емом "запоминающего устройства" механизма.

Подпись руководителя, ~~рецензента~~ А.Колмогоров

„8" мая 1952 г.

Оценка дипломной работы ____________________

Зав. кафедрой ____________________

„___" __________ 19___ г.

Отзыв на дипломную работу как руководителем, так и рецензентом представляется в 2 экземплярах, отпечатанных на машинке. Один экземпляр отзывов хранится в личном деле студента, второй экземпляр вместе с дипломной работой—на кафедре или в библиотеке факультета.

З. 688. Т. 2000

Московский ордена Ленина Государственный университет имени М. В. Ломоносова

Механико-математический **факультет**

кафедра ~~истории математических наук~~

ОТЗЫВ НА ДИПЛОМНУЮ РАБОТУ

студента У курса УСПЕНСКОГО В.А.

(фамилия и инициалы)

Тема Общее определение алгоритмической вычислимости и алгоритмической сводимости

~~Руководитель~~, рецензент профессор доктор физ.-матем. наук П.С.НОВИКОВ

(ученая степень, ученое звание, фамилия и инициалы)

В отзыве должны быть отмечены достоинства и недостатки дипломной работы.

Определение алгоритма не является определением в обычном для математики смысле этого слова. С его помощью не вводится в рассмотрение какое-нибудь новое понятие (и соответствующий ему термин). Оно должно выражать в точных математических терминах наиболее существенные свойства всякого математического алгоритма, позволяющего по заданию исходных данных, выполняя строго некоторые предписания, получить искомое. За последние годы предложен ряд определений алгоритма, которые все, однако, как убедительно показал автор, страдают рядом недостатков. Они либо носят очень специальный характер и не дают поэтому достаточных оснований не сомневаться в том, что ими отражены действительно наиболее существенные черты всякого математического алгоритма, либо же описывают не сам алгоритм (как процесс получения по входным данным искомого результата). В основу работы автором положено определение алгоритма, сообщенное ~~автору~~ ему А.Н. Колмогоровым. Глубоко и тонко проведенное В.А. Успенским исследование алгоритма Колмогорова показало, что это определение наилучшим образом выражает существо дела. Геометрический характер этого определения, использование в нем топологических комплексов делает его весьма наглядным, убедительным и общим.

В ходе проведенного им исследования алгоритма Колмогорова В.А. Успенскому пришлось решить ряд задач, относящихся к переводу алгорит-

мов в смысле других определений в алгоритм Колмогорова и доказательству рекурсивности последнего. ~~При этом~~ Таким образом автор не только дал методологически правильное – материалистическое – объяснение причин эквивалентности различных определений алгоритма, но и получил возможность включить их в единую теорию. Последнее оказывается особенно существенным во второй части работы, где автор дает – в терминах обобщенного на случай бесконечного числа входных данных алгоритма Колмогорова – общее определение алгоритмической сводимости вычисления значений одной функции к вычислению значений другой и доказывает ряд теорем о сводимости. Здесь особенно интересна теорема, дающая каноническое выражение для факта алгоритмической сводимости, аналогичное выражению всякой частично-рекурсивной функции через примитивно-рекурсивные функции и оператор μ.

Работа представляет значительный научный интерес и должна быть оценена как выдающаяся. Следует отметить также высокое качество оформления работы: работа написана простым, ясным и точным языком и читается с неослабевающим интересом. Ее, конечно, следует опубликовать.

Подпись ~~руководителя~~ рецензента П. Новиков

„——“ ———————— 19—— г.

Оценка дипломной работы отлично

Зав. кафедрой Р. Яновская

„——“ мая 1952 г.

Отзыв на дипломную работу как руководителем, так и рецензентом представляется в 2 экземплярах, отпечатанных на машинке. Один экземпляр отзывов хранится в личном деле студента, второй экземпляр вместе с дипломной работой—на кафедре или в библиотеке факультета.

З. 688. Т. 3000